VMIVME-DR11W-A

VMEbus-TO-DR11W INTERFACE

INSTRUCTION MANUAL

DOCUMENT NO. 500-000121-000 E

Revised 08 December 1993

VME MICROSYSTEMS INTERNATIONAL CORPORATION
12090 SOUTH MEMORIAL PARKWAY
HUNTSVILLE, AL 35803-3308
(205) 880-0444
1-800-322-3616

NOTICE

The information in this document has been carefully checked and is believed to be entirely reliable. While all reasonable efforts to ensure accuracy have been taken in the preparation of this manual, VMIC assumes no responsibility resulting from omissions or errors in this manual, or from the use of information contained herein.

VMIC reserves the right to make any changes, without notice, to this or any of VMIC's products to improve reliability, performance, function, or design.

VMIC does not assume any liability arising out of the application or use of any product or circuit described herein; nor does VMIC convey any license under its patent rights or the rights of others.

IMPORTANT NOTICE

CABLE REQUIREMENTS:

The VMIVME-DR11W-A requires that the following cabling guidelines be followed for proper operation and data integrity:

1. Flat-cable with an overall shield or ground plane (or superior cable) must be used. Mass-terminated standard flat-cable ***will*** cause data errors.

2. The cable shield must be grounded at both ends of the cable if a shielded cable is used. The cable grounds should have low impedance at high frequencies.

3. Macrolink interfaces for the Concurrent/Perkin-Elmer require a pinout reversal on the cables. When ordering specify macrolink cables.

DR11W COMPATIBILITY:

The VMIVME-DR11W-A is pinout compatible with DR11-W and DRV11-B type devices, with the exception of Function 1 now being a defined bit on the VMIVME-DR11W-A Board. It is ***not*** pinout compatible with DR11-B type devices and damage may result if it is cabled directly to a DR11-B type device.

Bit 6 (LONGWORD) of the Address Modifier Register (AMR) must be maintained to the proper state to assure proper operation of the associated 8-, 16-, or 32-bit transfers. Refer to Section 4.4.4.

RECORD OF REVISIONS

REVISION LETTER	DATE	PAGES INVOLVED	CHANGE NUMBER
A	03/21/91	Release per ECO	91-0051
B	10/07/91	Change per ECO	91-0228
C	02/10/92	Change per ECO	92-0022
D	02/22/93	Cover, pages iii, vi, viii, Sections 2, and 6, and Appendix A	92-0328
E	12/08/93	Correct Table 4.5-1	93-0913

VMIC
SAFETY SUMMARY

THE FOLLOWING GENERAL SAFETY PRECAUTIONS MUST BE OBSERVED DURING ALL PHASES OF THE OPERATION, SERVICE, AND REPAIR OF THIS PRODUCT. FAILURE TO COMPLY WITH THESE PRECAUTIONS OR WITH SPECIFIC WARNINGS ELSEWHERE IN THIS MANUAL VIOLATES SAFETY STANDARDS OF DESIGN, MANUFACTURE, AND INTENDED USE OF THIS PRODUCT. VME MICROSYSTEMS INTERNATIONAL CORPORATION ASSUMES NO LIABILITY FOR THE CUSTOMER'S FAILURE TO COMPLY WITH THESE REQUIREMENTS.

GROUND THE SYSTEM

To minimize shock hazard, the chassis and system cabinet must be connected to an electrical ground. A three-conductor AC power cable should be used. The power cable must either be plugged into an approved three-contact electrical outlet or used with a three-contact to two-contact adapter with the grounding wire (green) firmly connected to an electrical ground (safety ground) at the power outlet.

DO NOT OPERATE IN AN EXPLOSIVE ATMOSPHERE

Do not operate the system in the presence of flammable gases or fumes. Operation of any electrical system in such an environment constitutes a definite safety hazard.

KEEP AWAY FROM LIVE CIRCUITS

Operating personnel must not remove product covers. Component replacement and internal adjustments must be made by qualified maintenance personnel. Do not replace components with power cable connected. Under certain conditions, dangerous voltages may exist even with the power cable removed. To avoid injuries, always disconnect power and discharge circuits before touching them.

DO NOT SERVICE OR ADJUST ALONE

Do not attempt internal service or adjustment unless another person, capable of rendering first aid and resuscitation, is present.

DO NOT SUBSTITUTE PARTS OR MODIFY SYSTEM

Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification to the product. Return the product to VME Microsystems International Corporation for service and repair to ensure that safety features are maintained.

DANGEROUS PROCEDURE WARNINGS

Warnings, such as the example below, precede only potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed.

WARNING

DANGEROUS VOLTAGES, CAPABLE OF CAUSING DEATH, ARE PRESENT IN THIS SYSTEM. USE EXTREME CAUTION WHEN HANDLING, TESTING, AND ADJUSTING.

SAFETY SYMBOLS

GENERAL DEFINITIONS OF SAFETY SYMBOLS USED IN THIS MANUAL

⚠ Instruction manual symbol: the product is marked with this symbol when it is necessary for the user to refer to the instruction manual in order to protect against damage to the system.

ϟ Indicates dangerous voltage (terminals fed from the interior by voltage exceeding 1000 volts are so marked).

⏚ OR (⏚) Protective conductor terminal. For protection against electrical shock in case of a fault. Used with field wiring terminals to indicate the terminal which must be connected to ground before operating equipment.

(⏚) Low-noise or noiseless, clean ground (earth) terminal. Used for a signal common, as well as providing protection against electrical shock in case of a fault. Before operating the equipment, terminal marked with this symbol must be connected to ground in the manner described in the installation (operation) manual.

⊥ OR ⊥ Frame or chassis terminal. A connection to the frame (chassis) of the equipment which normally includes all exposed metal structures.

∿ Alternating current (power line).

⎓ Direct current (power line).

≂ Alternating or direct current (power line).

WARNING The WARNING sign denotes a hazard. It calls attention to a procedure, a practice, a condition, or the like, which, if not correctly performed or adhered to, could result in injury or death to personnel.

```
***************
*   CAUTION   *
***************
```

The CAUTION sign denotes a hazard. It calls attention to an operating a procedure, a practice, a condition, or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the system.

NOTE: The NOTE sign denotes important information. It calls attention to a procedure, a practice, a condition or the like, which is essential to highlight.

VMIVME-DR11W-A
VMEbus-TO-DR11W INTERFACE

TABLE OF CONTENTS

Page

SECTION 1. INTRODUCTION

SECTION 2. PHYSICAL DESCRIPTION AND SPECIFICATIONS

SECTION 3. THEORY OF OPERATION

SECTION 4. PROGRAMMING

TABLE OF CONTENTS (Continued)

SECTION 4. PROGRAMMING (Concluded)

SECTION 5. CONFIGURATION AND INSTALLATION

TABLE OF CONTENTS (Continued)

Page

SECTION 5. CONFIGURATION AND INSTALLATION (Concluded)

SECTION 6. MAINTENANCE WARRANTY

LIST OF FIGURES

Figure Page

TABLE OF CONTENTS (Continued)

LIST OF FIGURES (Continued)

Figure Page

TABLE OF CONTENTS (Continued)

LIST OF FIGURES (Concluded)

Figure Page

LIST OF TABLES

Table Page

APPENDICES

SECTION 1

INTRODUCTION

1.1 INTRODUCTION

The VMIVME-DR11W-A is a high performance VMEbus 32-bit Direct Memory Access (DMA) interface that forms a 16-bit parallel interface between the VMEbus and Digital Equipment Corporation (DEC) DR11-W and DRV11-WA type devices, and with the DR11W CPU links. A photograph of the VMIVME-DR11W-A printed circuit assembly is shown in Figure 1.1-1. VMIVME-DR11W-A features are as follows:

a. DR11W and DRV11-WA compatible I/O connection
b. 32-bit DMA data transfers
c. 32-bit addressing
d. VMEbus (IEEE P1014) compatible
e. Software-selectable byte and word swapping
f. Watchdog timers prevent transfer lockout on VMEbus and cables
g. Front panel Fail LED
h. Supports off-line Built-in-Test plus single board loopback testing (with test cable)
i. Meets VMEbus Spec. C.1 - compatible address pipelining
j. Fully programmable operation (includes selection of 16- or 32-bit transfer, burst mode, etc.)
k. Fully programmable interrupt vectors and levels
l. Programmable VMEbus address modifiers
m. Jumper-selectable VME standard short I/O address space for on-board registers
n. Software compatible with VMIC's DR11W and DR11W-485 boards
o. Two VMIVME-DR11W-As form a VMEbus-to-VMEbus link

1.2 FUNCTIONAL DESCRIPTION

The VMIVME-DR11W-A provides a high performance, 16-bit DMA Link between the VMEbus and DEC DR11-W compatible devices. The VMIVME-DR11W-A interfaces may also be connected back-to-back to form a VMEbus-to-VMEbus DMA Link (see Figures 1.2-1 and 1.2-2).

The VMIVME-DR11W-A incorporates a SCB68430 DMA controller device to perform its DMA transfers, a MC68153 Bus Interrupter Module (BIM) to implement a VMEbus interrupter (for the attention and DMA completion interrupts), and it also contains logic to allow the VMIVME-DR11W-A to generate an attention interrupt to the external device.

MDR11W-A/F1.1-1

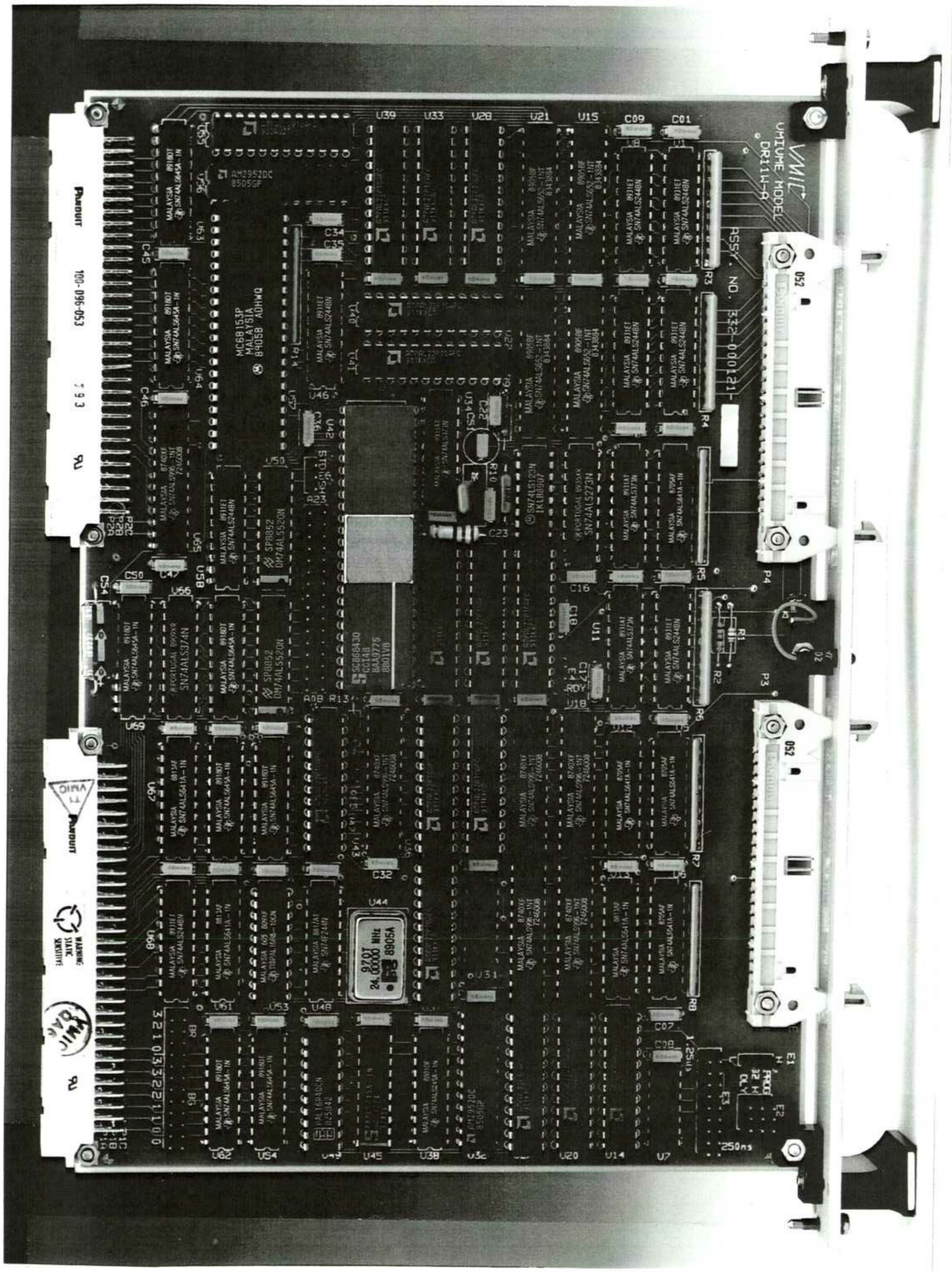
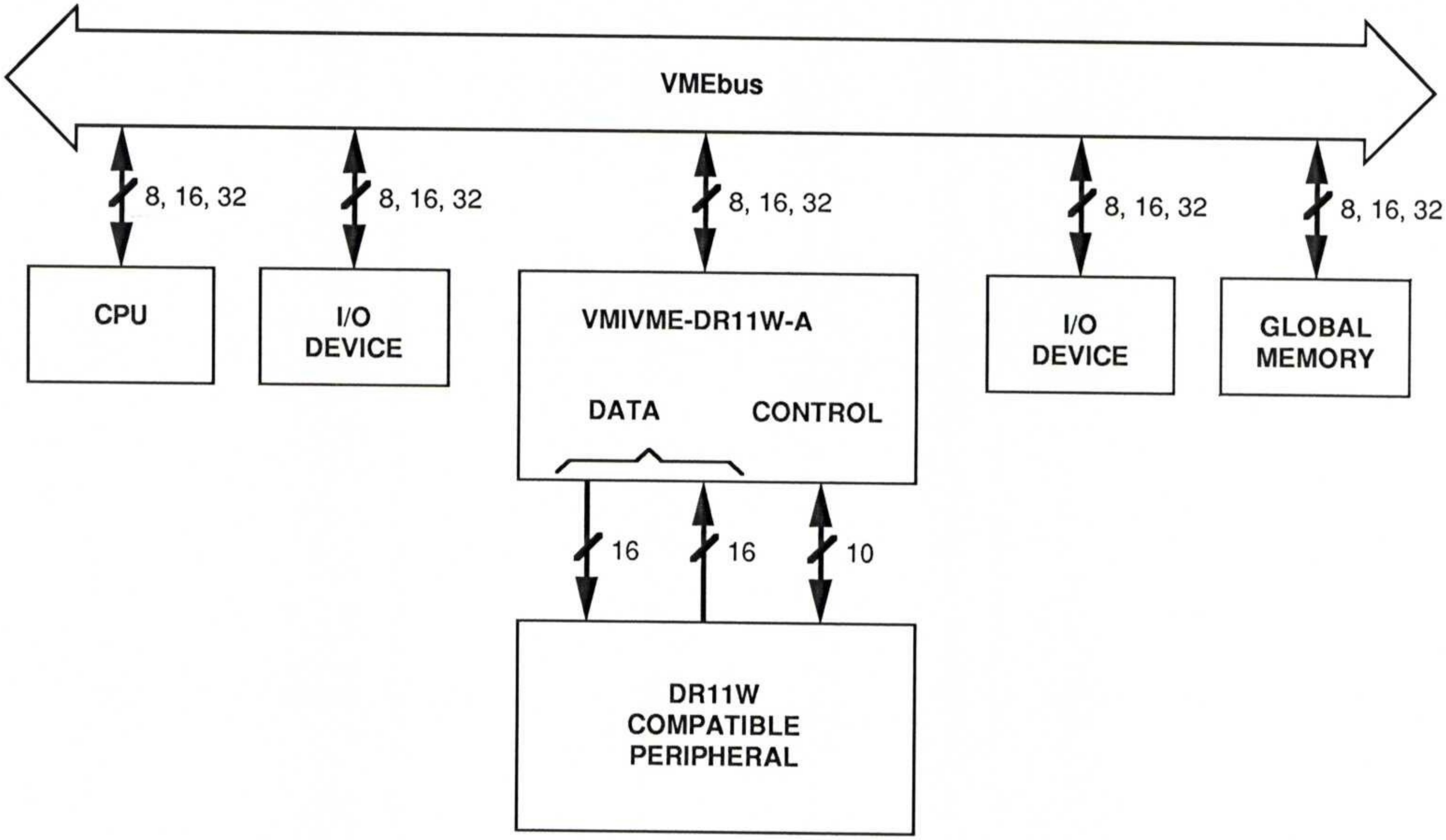
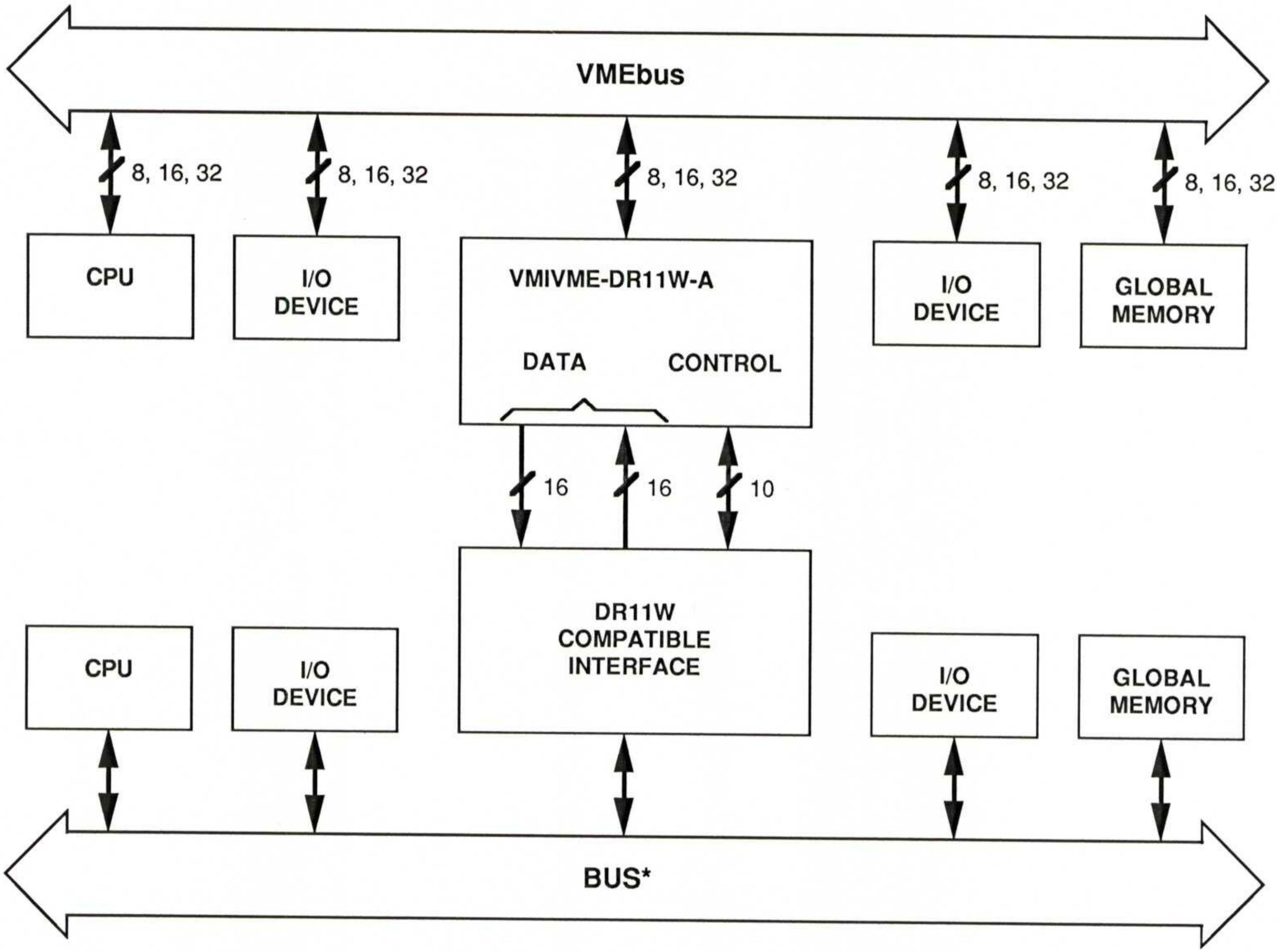


Figure 1.1-1. Printed Circuit Assembly Photograph



MDR11W-A/F1.2-1

Figure 1.2-1. VMIVME-DR11W-A Connection-to-User Device



*DEC, Concurrent, Data General, Harris Night Hawk, Prime, IBM PC/AT, Masscomp, Sun Workstations, or other VMEbus based machines.

MDR11W-A/F1.2-2

Figure 1.2-2. VMIVME-DR11W-A CPU-to-CPU Link

1.3 COMPATIBILITY

VMIVME-DR11W compatibility interfaces for the computers listed in Table 1.3-1 are available from the companies indicated for each computer.

Table 1.3-1. VMIVME-DR11W-A Compatible Host Computer Interface Sources

COMPUTER	COMPANY	PHONE NUMBER
DEC VAX, PDP-11 LSI-11	Digital Equipment Corp.	1-800-DIGITAL Continental Blvd. Merrimack, NH 03054
CONCURRENT/ PERKIN-ELMER	Macrolink**	(714) 634-8080
HARRIS H60-H100	Harris	(404) 256-4000
ENCORE CONCEPT 32 Series*	Encore	(305) 587-2900
IBM PC/AT	Omnicomp Graphics***	(713) 464-2990 Houston, TX

*Not directly VMIVME-DR11W compatible - requires simple modification of MPCI Model 8020 IOP interface board (contact Encore for VMIVME-DR11W modification kit). VMIC has a 32-bit Encore HSD-to-VMEbus interface.

**Macrolink interfaces require a cable pinout reversal. When ordering cables specify the macrolink cables.

***Requires Omnicomp interprocessor link driver with soft start.

MDR11W-A/T1.3-1

NOTICE

THE INFORMATION IN TABLE 1.3-1 IS OFFERED AS A HELPFUL SUGGESTION. IT IS INTENDED AS A GUIDE. VMIC DOES NOT GUARANTEE FAVORABLE RESULTS NOR DOES IT ASSUME ANY LIABILITY IN CONNECTION WITH ITS USE. NO STATEMENT OR RECOMMENDATION IS INTENDED BY VMIC FOR THE USE OF THESE PRODUCTS. BEFORE USING THE PRODUCTS LISTED, THE USER SHALL DETERMINE THE SUITABILITY OF THE PRODUCT(S) FOR HIS INTENDED USE, AND THE USER ASSUMES ALL RISK AND LIABILITY IN CONNECTION WITH THEIR USE.

1.4 REFERENCE MATERIAL LIST

The reader should refer to "The VMEbus Specification" for a detailed explanation of VMEbus. "The VMEbus Specification is available from the following source:

VITA
VMEbus International Trade Association
10229 N. Scottsdale Road
Scottsdale, AZ 85253
(602) 951-8866

Additional information concerning the DR11W protocol is contained in the "DR11-W Direct Memory Interface Module (No. EK-DR11W-UG-002)". This document is available from the following source:

Digital Equipment Corporation
146 Main St.
Maynard, MA 01754-2751
(617) 897-5111

Specific information concerning the SCB68430 may be obtained from the following source:

Signetics Corporation
811 E. Arques Ave.
P.O. Box 3409
Sunnyvale, CA 94088-3409
(408) 991-2000

A detailed specification is provided in Appendix B.

SECTION 2

PHYSICAL DESCRIPTION AND SPECIFICATIONS

REFER TO 800-000121-000 SPECIFICATION

SECTION 3

THEORY OF OPERATION

3.1 BLOCK DIAGRAMS

As shown in the functional block diagram (see Figure 3.1-1) and in the interconnection diagrams (Figures 3.1-2 and 3.1-3) the VMIVME-DR11W-A interface provides for the exchange of program controlled interrupts and status as well as two 16-bit data buses with handshake signals. The function of these interface signals are described in Section 3.4. The handshake sequencing of the signals are depicted in Figures 3.1-4, 3.1-5, and 3.1-6 for the 32-bit transfer mode, and in Figures 3.1-7, 3.1-8, and 3.1-9 for the 16-bit transfer mode.

Although the VMIVME-DR11W-A interface was developed for interconnection to DR11-W compatible devices, it also performs DMA transfers in conjunction with the DRV-11WA interface (see Figure 3.1-3).

3.2 OPERATIONAL OVERVIEW

The VMIVME-DR11W-A has two modes of operation, INTERPROCESSOR LINK MODE (back-to-back configuration), and EXTERNAL DEVICE MODE. In the INTERPROCESSOR LINK MODE, the VMIVME-DR11W-A communicates with another DR11-W compatible computer interface. In the EXTERNAL DEVICE MODE, the VMIVME-DR11W-A communicates with a user's device such as a disk drive system, tape drive, etc.

To operate in either mode, a transfer protocol (see Section 3.3) must be established. Once this is determined, the VMIVME-DR11W-A may be programmed as detailed in Section 4.1. One difference in the operation of this device versus other DR11-W implementations is that the bus transfer type (word/byte and direction) is determined before the start of any data transfers and not on a cycle-by-cycle basis.

When two DR11W compatible boards are connected back-to-back the two cables are crossed over so that one board drives a control signal and the other board receives it. Thus, there is no need for a "master/slave mode".

3.3 TRANSFER PROTOCOL

In order for DMA transfers to take place the DMA interfaces must be set up at both ends of the link. Therefore, if a CPU on one bus wants to transfer data, a

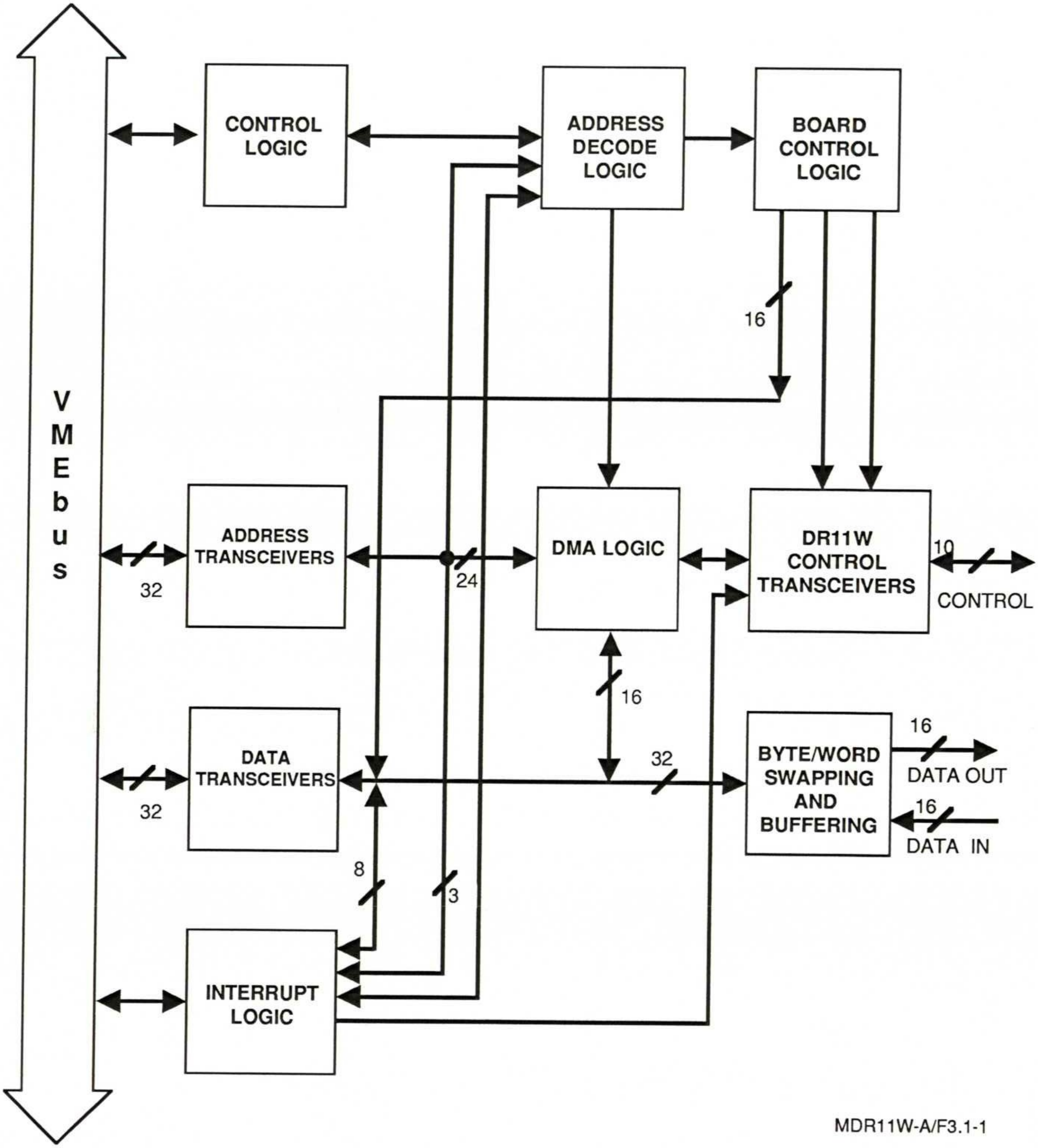
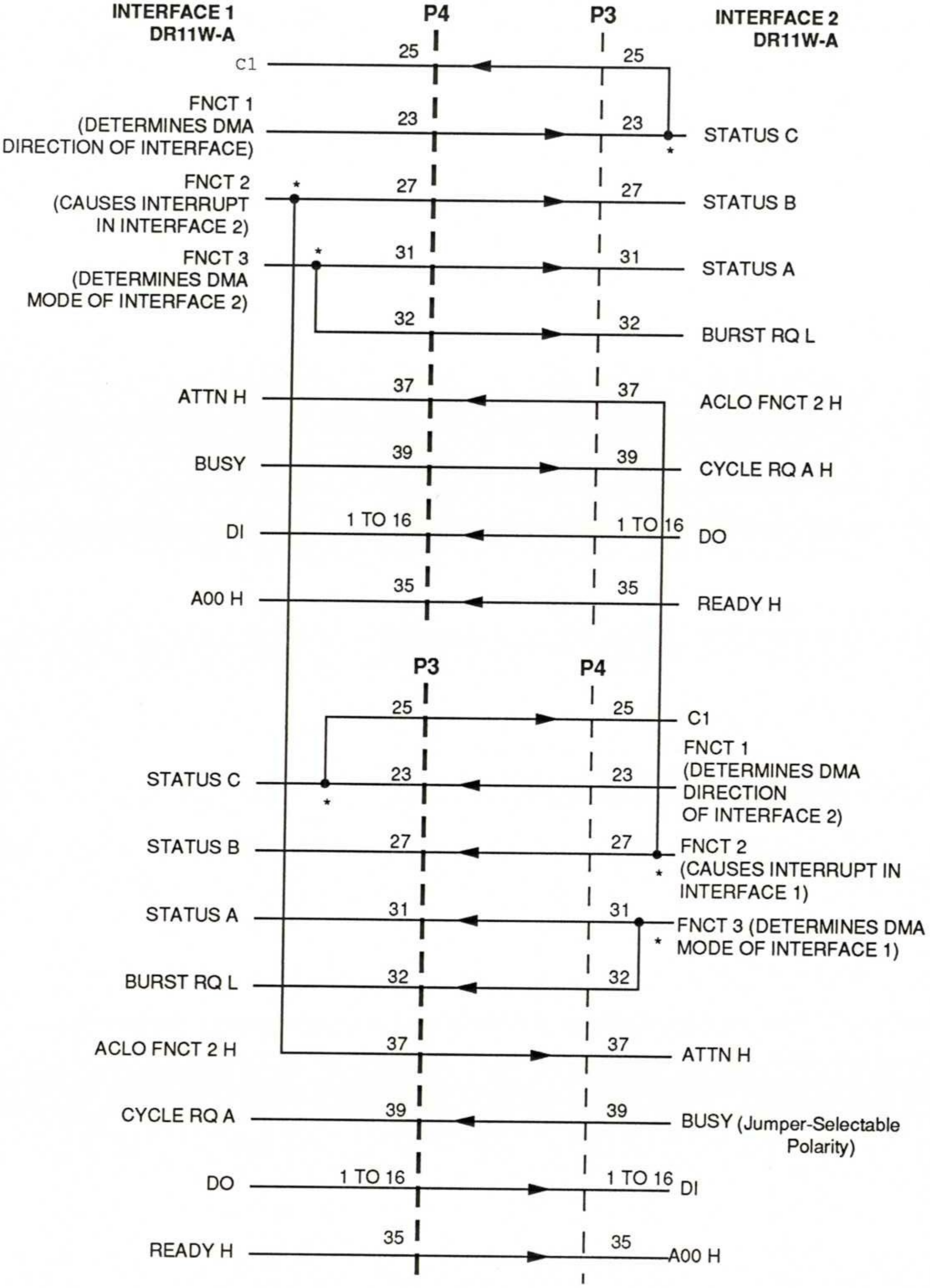


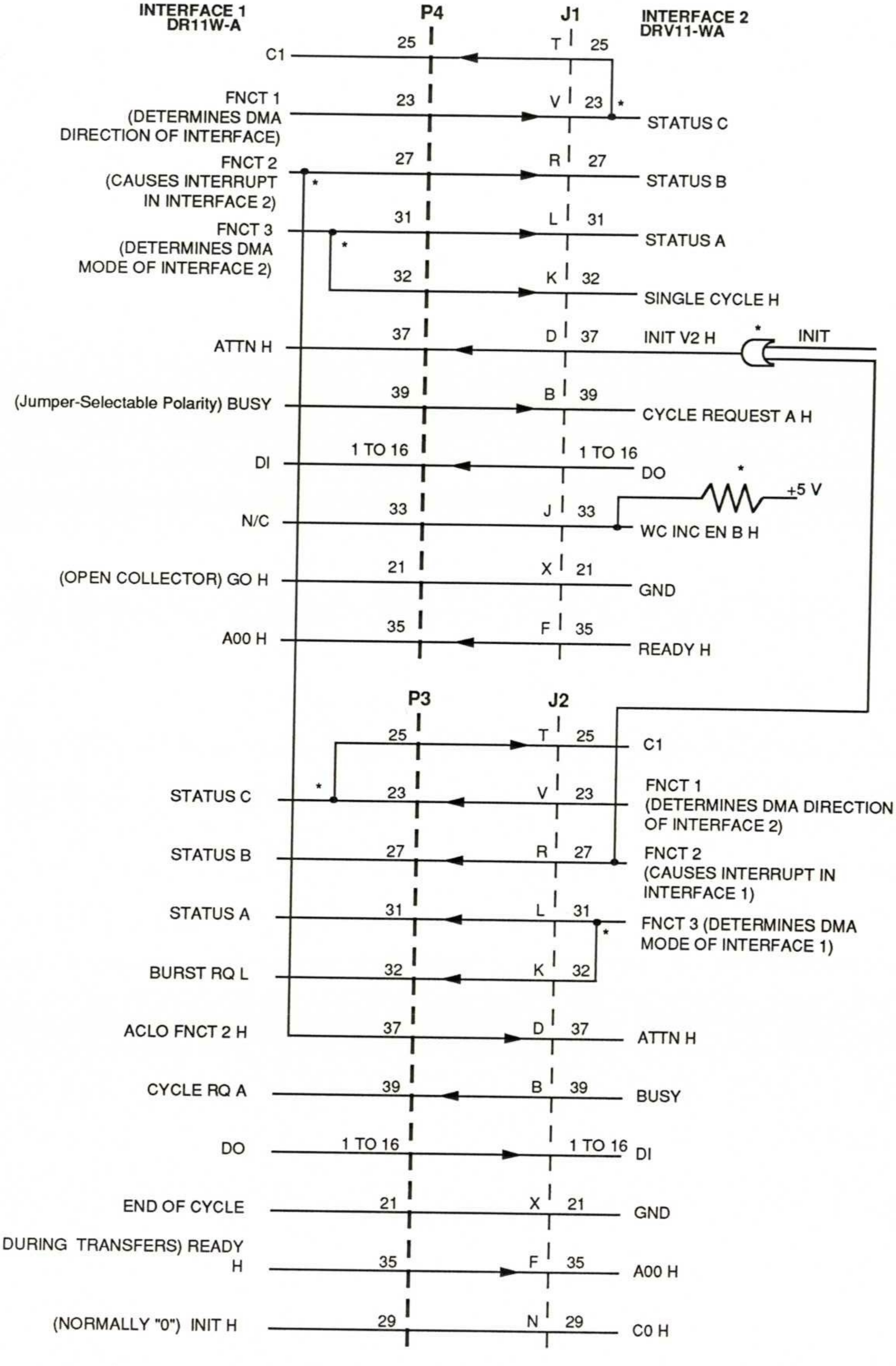
Figure 3.1-1. VMIVME-DR11W-A Functional Block Diagram



*Connection is hardwired on PC board.

MDR11W-A/F3.1-2

Figure 3.1-2. Interface Interconnection for Two VMIVME-DR11W-As in Back-to-Back Configuration



*All components and connections shown are included in the standard products.

MDR11W-A/F3.1-3

Figure 3.1-3. Interface Connection to a DRV11-WA

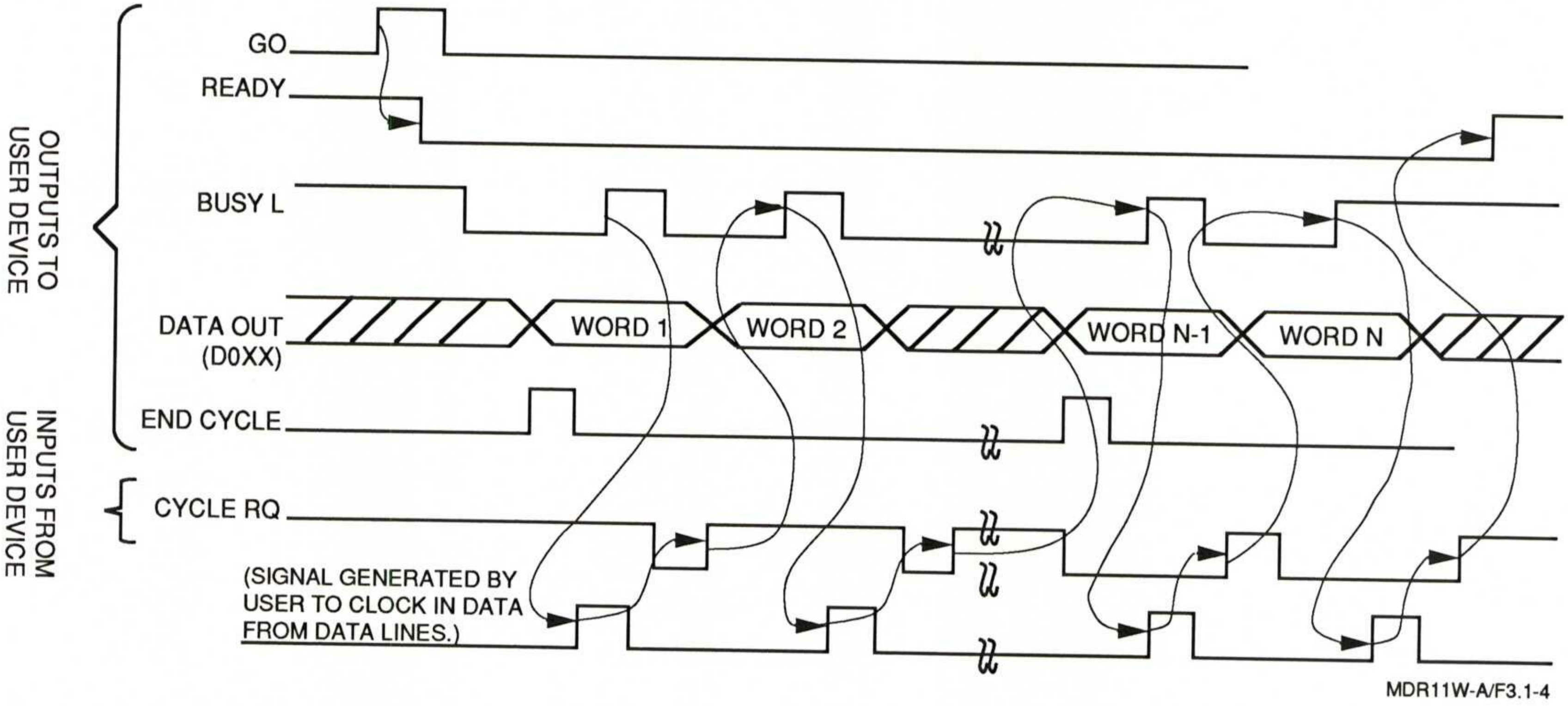


Figure 3.1-4. Timing/Handshake Sequencing 32-Bit Data Transfers from the VMIVME-DR11W-A to the User Device

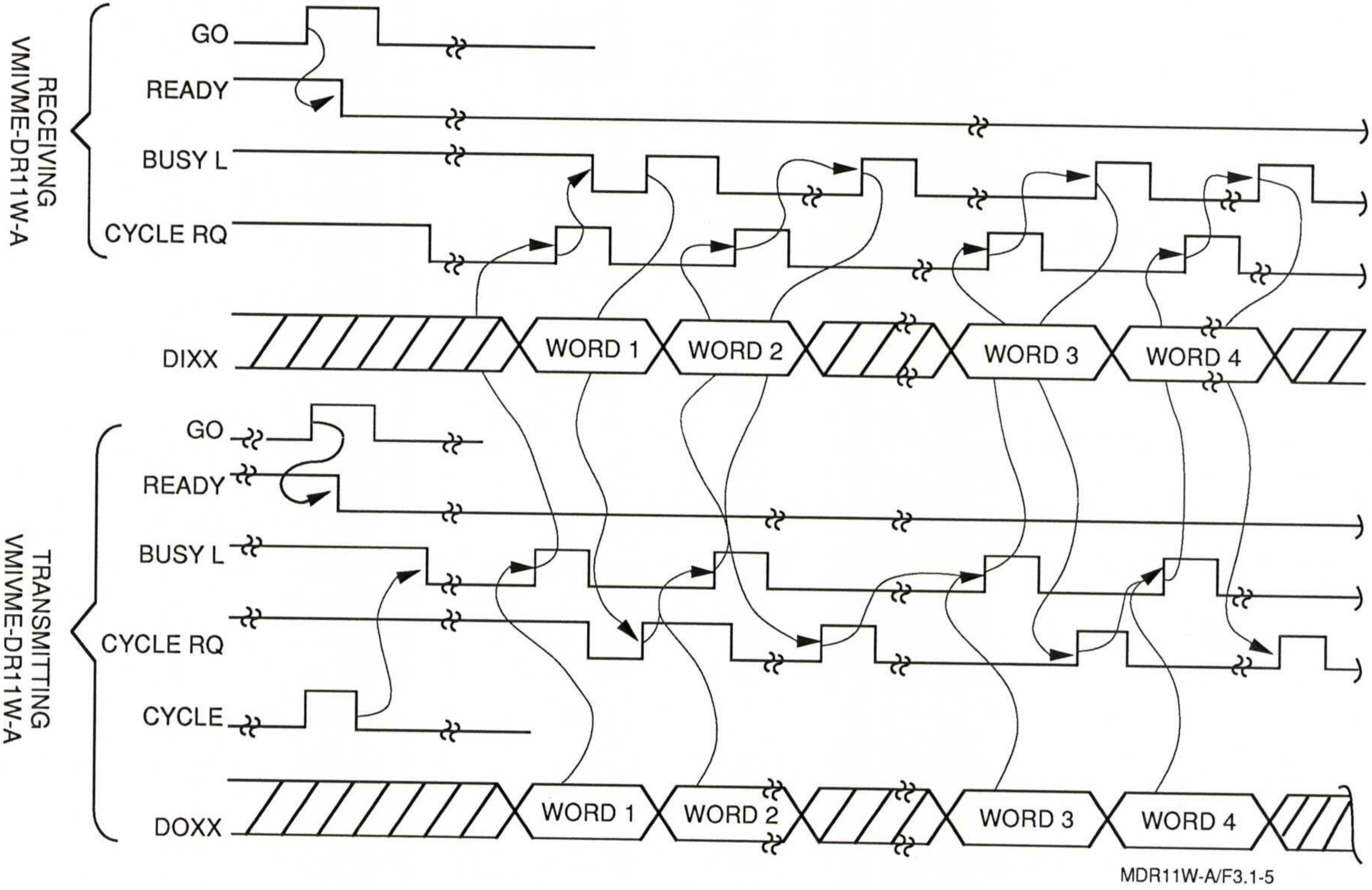


Figure 3.1-5. Back-to-Back 32-Bit DMA Transfers Configuration (Interprocessor Mode)

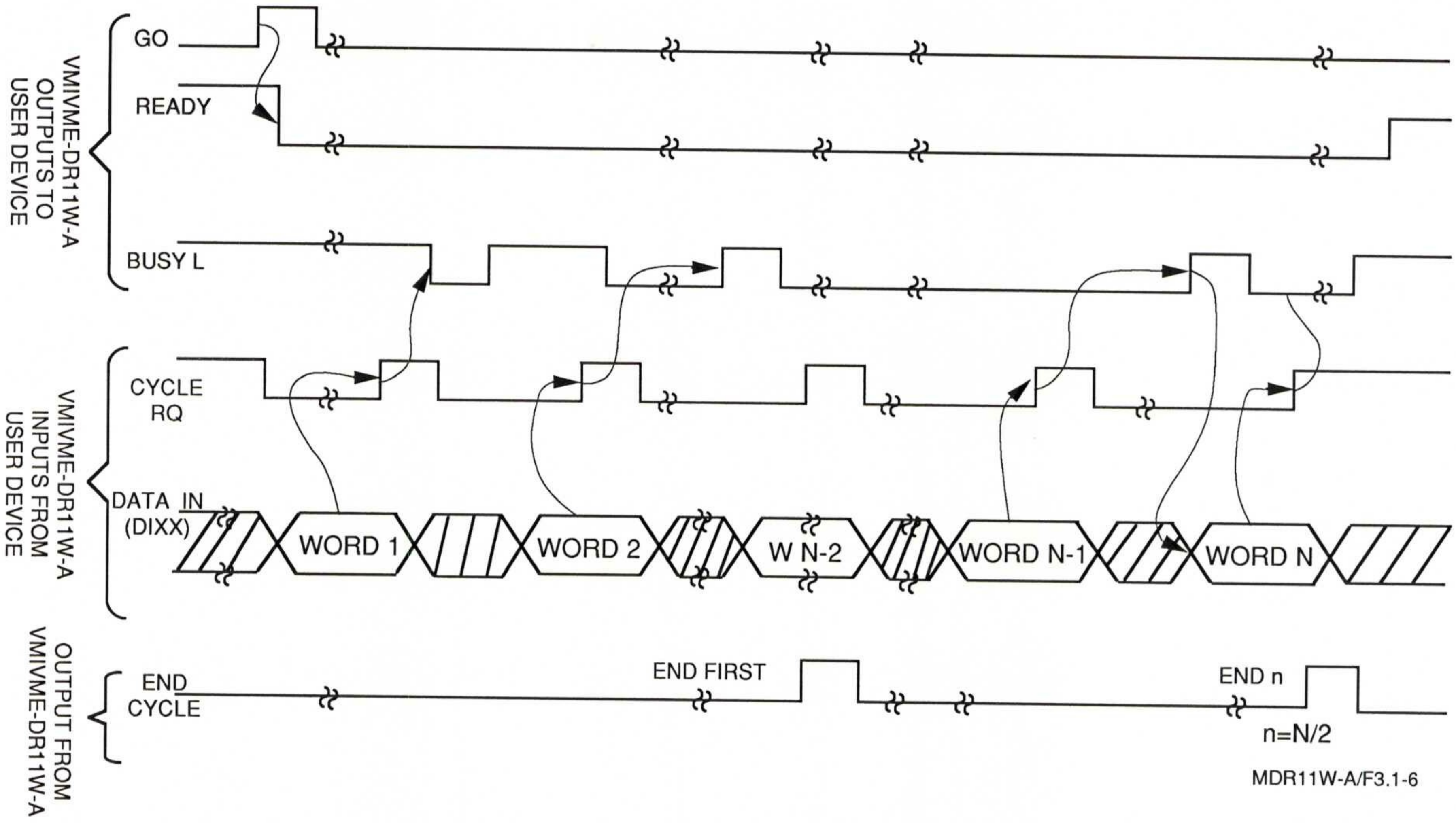
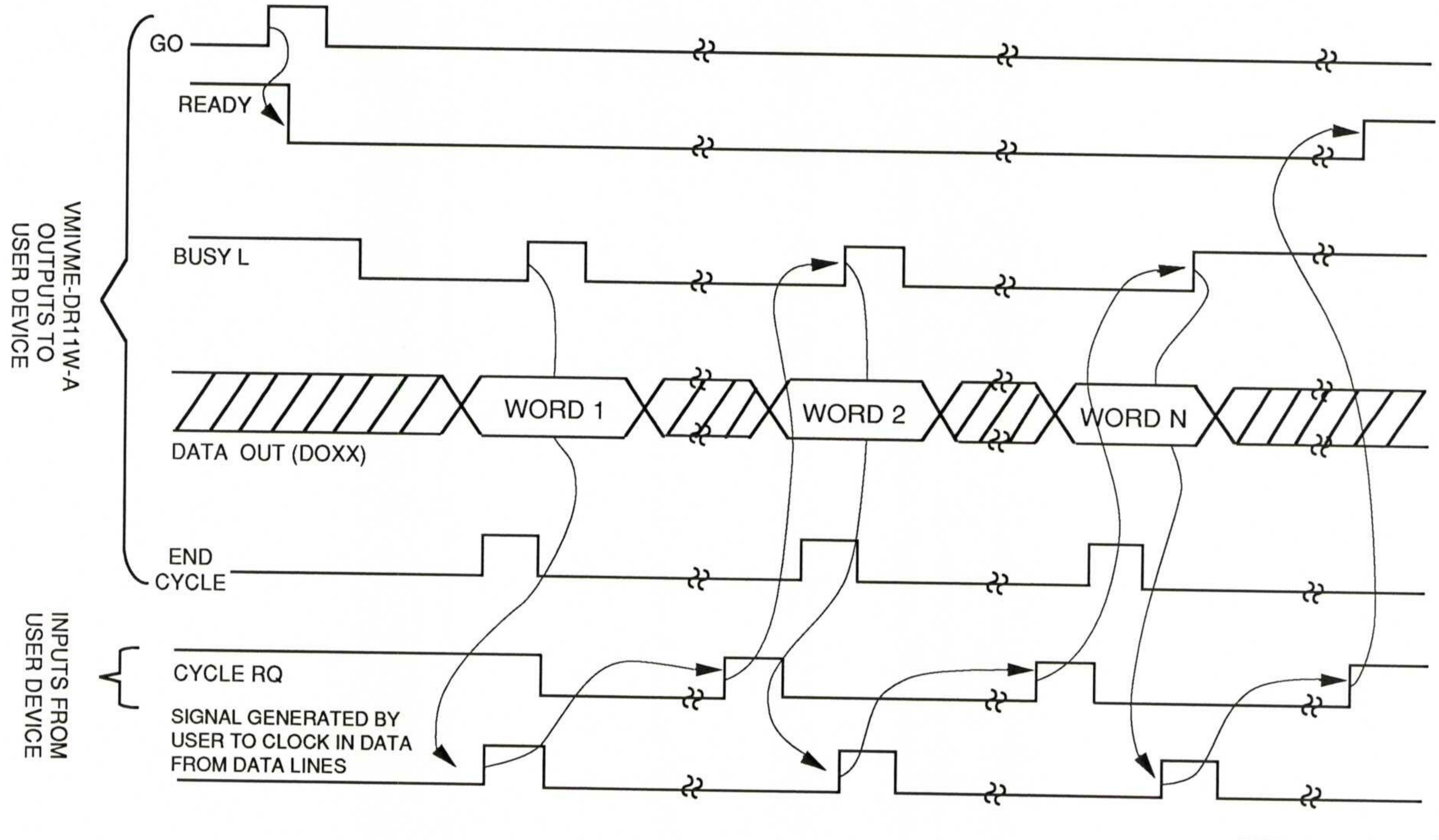


Figure 3.1-6. Diagram for 32-Bit Data Transfers from the User Device to the VMIVME-DR11W-A



MDR11W-A/F3.1-7

Figure 3.1-7. Timing/Handshake Sequencing 16-Bit Data Transfers from the VMIVME-DR11W-A to the User Device

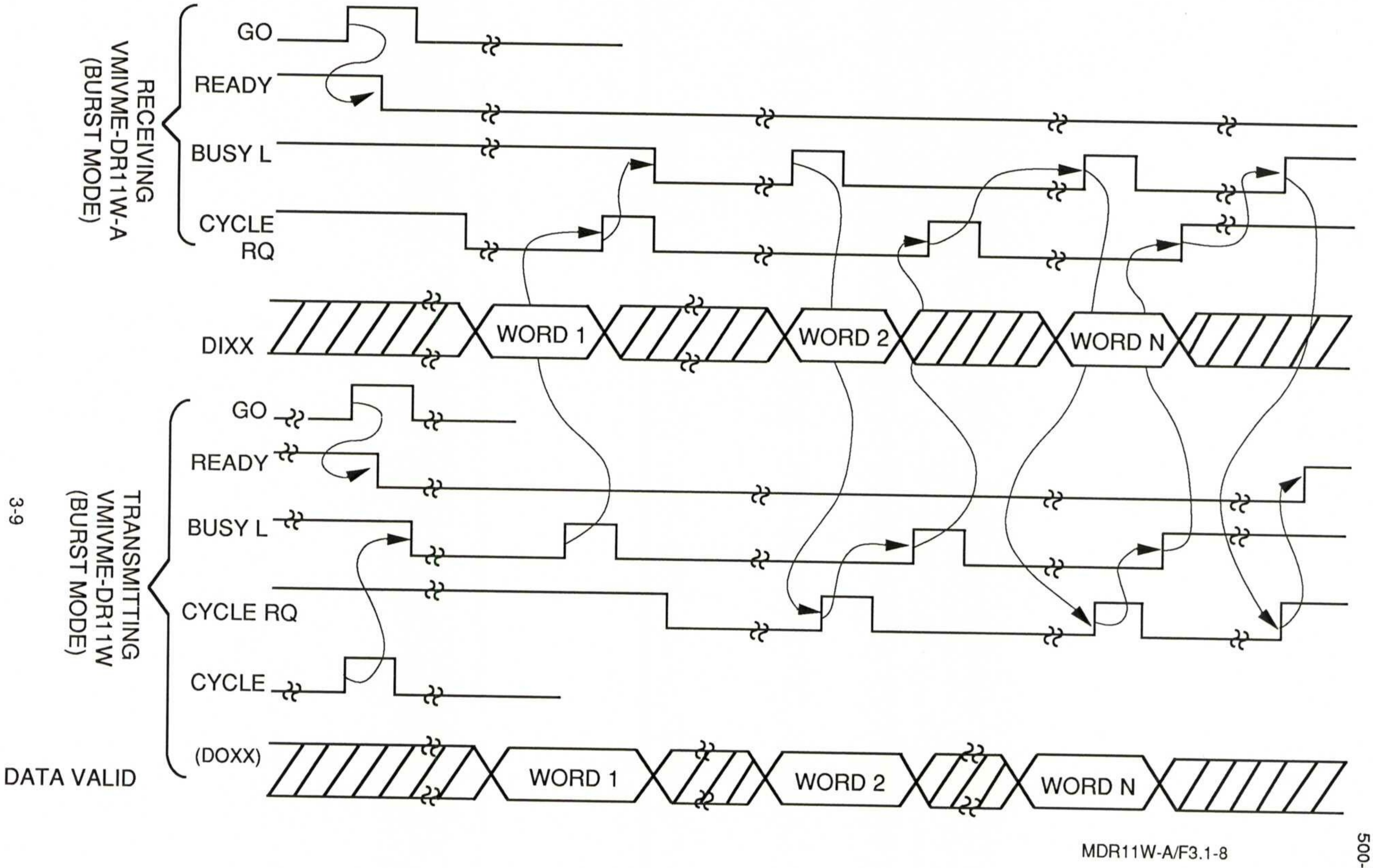


Figure 3.1-8. Back-to-Back 16-Bit DMA Transfer Configuration (Interprocessor Mode)

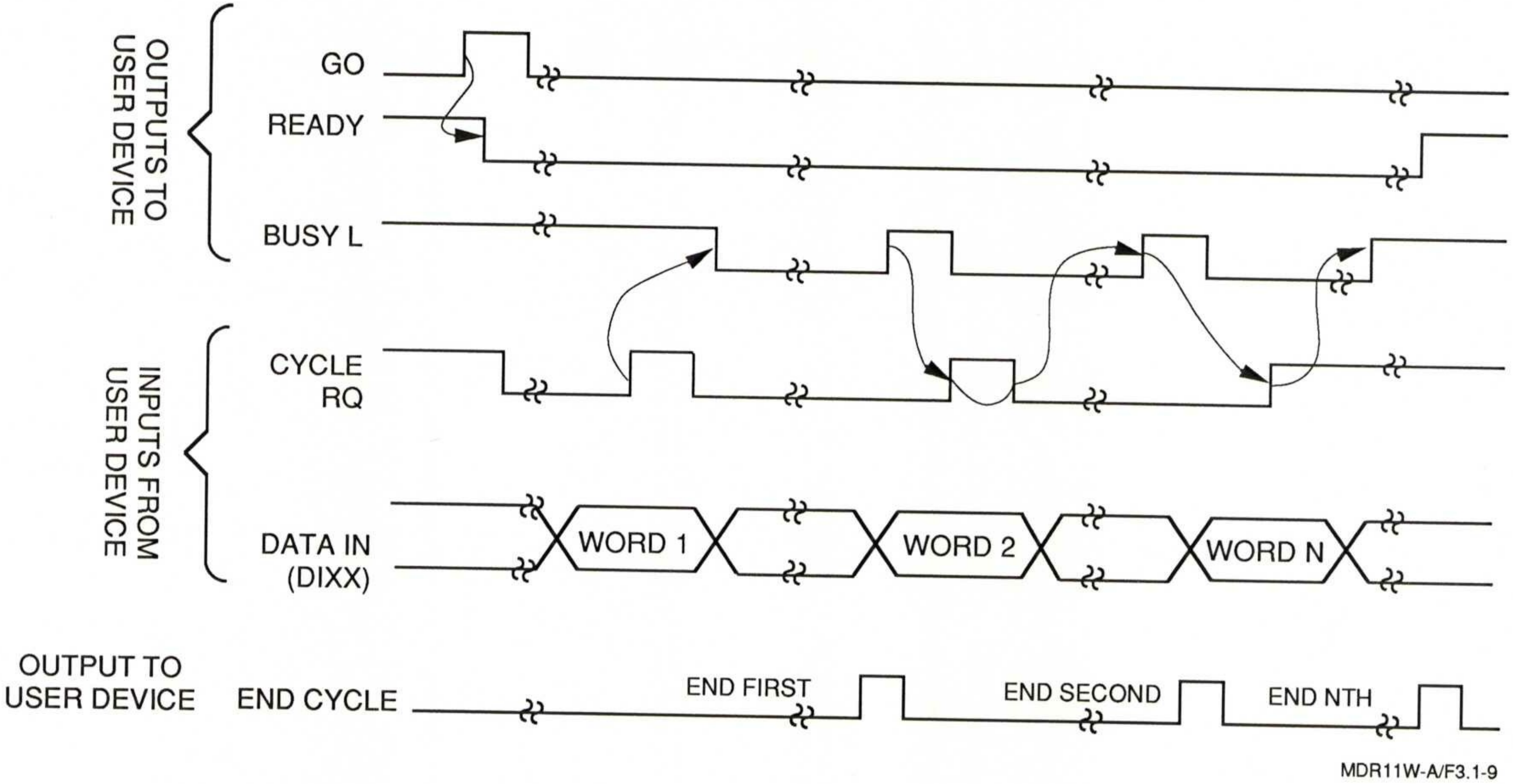


Figure 3.1-9. Diagram for 16-Bit Data Transfers from the User Device to the VMIVME-DR11W-A

second CPU on the second bus must set up the DMA interface on that second bus to the selected memory address, word count, and direction of transfer.

The sequence of operations necessary for two CPUs on two separate buses to control data transfers is referred to as the transfer protocol. There are several types of protocols that may be used with the VMIVME-DR11W-A.

One protocol type consists of the following sequences:

a. The CPU in bus one interrupts the CPU on the other bus via the "ATTENTION" interrupt.

b. Both DMA interfaces are set up to transfer a control data block. The control data block contains information required by the second system to set up the other VMIVME-DR11W-A (such as transfer counts, transfer direction, etc.).

c. The control data block is then used to set up the DMA interface in the second bus. This protocol preselects the size and direction of transfer of the control block.

d. Transfer of data then occurs after both boards have set "GO," both boards have sensed that "OTHER BOARDS READY" from the A00H cable signal (see Section 5.3.5) is at the proper level, and a "BUSY" cable signal is generated by one of the two boards (request cycle).

A second protocol type consists of using predetermined transfer block sizes and sequences.

A third protocol type consists of sending the DMA set-up data via a serial communication link.

3.4 DATA TRANSFER DESCRIPTION

When the VMIVME-DR11W-A transfers individual data words to or from an external device, it becomes the VMEbus master and transfers the data via DMA transfers to or from memory. It asserts the VMEbus address, control lines, address modifier, and sends or receives data. The data transfers are controlled by a SCB68430 DMA interface chip. Handshake signals to/from the SCB68430 are converted to/from DR11W compatible handshake signals.

3.5 VME-TO-VME LINK

Two VMIVME-DR11W-A boards may be connected back-to-back to create a DMA link between two VMEbus chassis assemblies. In this configuration, the two boards are connected so their data buses and control signals are cross coupled which allows them to transfer data back and forth under handshake control. The cross coupling is shown in Figure 3.1-2.

3.6 USER INTERFACE SIGNAL FUNCTION DESCRIPTION

The following paragraphs describe the standard DR11W signals used by the VMIVME-DR11W-A.

DO00 TO DO15 - DATA OUTPUT BUS. The data is positive true and is transmitted by bus transceivers. Each line is terminated by a 180/390 Ω network.

CYCLE REQUEST A - CYCLE REQUEST B. These two inputs are used to request a DMA cycle from the user's device. These two signals are "ORed" together.

END CYCLE. This signal indicates the end of a DMA cycle to the user's device.

STATUS A - STATUS BIT A. This bit is a user-defined status bit from the user's device to the VMIVME-DR11W-A. In the back-to-back configuration, this bit indicates that single cycle transfers should be performed when it is asserted.

STATUS B - STATUS BIT B. This bit is a user-defined status bit from the user's device to the VMIVME-DR11W-A. In the back-to-back configuration, STATUS B indicates the state of the ATTN interrupt bit.

STATUS C - STATUS BIT C. This signal is a user-defined status bit from the user's device to the VMIVME-DR11W-A. In the back-to-back configuration, STATUS C indicates the direction of the DMA transfer.

INIT - INITIALIZE. This signal is output from the VMIVME-DR11W-A to indicate a power ON reset or other reset has occurred.

BURST RQ - BURST REQUEST. This signal is an input to the VMIVME-DR11W-A and when asserted, indicates that burst transfers should be performed. This bit is negative true.

READY. This output signal indicates that the board is ready and does not have transfers in progress. This signal is de-asserted by GO.

ACLO FNCT 2 - AC LOW OR FUNCTION2. This bit is the same as Function 2 on the VMIVME-DR11W-A.

DI00 TO DI15 - DATA INPUT BUS. The data is positive true and is received by bus transceivers. Each line is terminated by a 180/390 Ω network.

GO. This bit is output by the VMIVME-DR11W-A to indicate the start of a DMA transfer block.

FNCT 1 - FUNCTION 1. Board Control Register (BCR) bit 05 (defined) is output by the VMIVME-DR11W-A. In the back-to-back configuration this bit controls the DMA direction of the VMIVME-DR11W-A and drives C1.

FNCT 2 - FUNCTION 2. Board Control Register (BCR) bit 06 (user-defined) is output from the VMIVME-DR11W-A. In the back-to-back configuration, this bit drives STATUS B (and ATTN) on the other board.

FNCT 3 - FUNCTION 3. Board Control Register (BCR) bit 07 (user-defined) is output from the VMIVME-DR11W-A. In the back-to-back configuration, this bit drives STATUS A.

C1 - CONTROL 1. This signal is an input to the VMIVME-DR11W-A and controls the direction of DMA transfer. In the back-to-back configuration, it is driven by FNCT 1. A logic "one" on C1 causes the VMIVME-DR11W-A to receive data while a logic "zero" causes it to transmit.

ATTN - ATTENTION. This signal is an input to the VMIVME-DR11W-A. It indicates that the user's device has issued attention interrupt request and causes an interrupt to the VMEbus system on the rising edge.

BUSY. This signal is an output from the VMIVME-DR11W-A and indicates that a VMEbus cycle is in progress. The polarity of this signal is jumper-selectable (Busy H or Busy L). In the BACK-TO-BACK MODE, Busy L must be selected since it is connected to the cycle request on the other DR11W board.

SECTION 4

PROGRAMMING

4.1 PROGRAMMING OVERVIEW

The VMIVME-DR11W-A has three register groups that must be programmed for proper operation. These register groups are associated with the SCB68430 DMA controller module, the MC68153 Bus Interrupter Module (BIM), and the on-board Control Registers. The on-board Control Registers are provided to allow the user the capability of programming the address modifiers, to allow control of the DMA controller and the BIM, to allow interface with the on-board error detection logic, to implement the VMIVME-DR11W-A on-board master control functions, and to communicate with the external VMIVME-DR11W-A compatible device.

The register address map showing the register names and addresses is shown in Table 4.1-1. The register formats are described in Sections 4.2, 4.3, and 4.4, where the Individual Register bit assignments are detailed.

4.2 PROGRAMMING THE SCB68430 DMAI

Programming the SCB68430 DMAI requires interface with most of the 14 on-chip registers that are described in this section. Table 4.2-1 details the DMAI register formats for the DMAI register bit descriptions that follow. Refer to the SCB68430 DMAI data sheet contained in Appendix B for additional detailed operation and programming information for the DMAI.

4.2.1 Channel Status Register/Channel Error Register

The Channel Status Register (CSR) and Channel Error Register (CER) are used together to determine status and error conditions related to the DMA controller.

4.2.1.1 CSR Bit Definitions

BIT 15 - CHANNEL OPERATION COMPLETE*. This bit will be set when the DMA operation is complete. If interrupts are enabled, an interrupt will be generated at this time.

***NOTICE**

IF SET, CSR BIT 15 AND CSR BIT 12 MUST BE CLEARED BEFORE ANOTHER DMA OPERATION CAN BE STARTED. THESE BITS MAY BE CLEARED BY WRITING A "ONE" TO EACH BIT THAT IS SET. THE FOLLOWING 68000 INSTRUCTION WILL ACCOMPLISH THIS WITHOUT TESTING EACH OF THE BITS. *THE OTHER BITS IN THE CSR ARE UNAFFECTED BY A WRITE.*

```
MOVE. B        CSR,CSR
```

Table 4.1-1. VMIVME-DR11W-A Register Address Map

HEX ADDRESS	ACRONYM	REGISTER NAME	PHYSICAL LOCATION
00	CSR	Channel Status Register	SCB68430 DMAI, U42, Sheet 3
01	CER	Channel Error Register	SCB68430 DMAI, U42, Sheet 3
04	DCR	Device Control Register	SCB68430 DMAI, U42, Sheet 3
05	OCR	Operation Control Register	SCB68430 DMAI, U42, Sheet 3
06	SCR*	Sequence Control Register	SCB68430 DMAI, U42, Sheet 3
07	CCR	Channel Control Register	SCB68430 DMAI, U42, Sheet 3
0A	MTCH	Memory Transfer Count High	SCB68430 DMAI, U42, Sheet 3
0B	MTCL	Memory Transfer Count Low	SCB68430 DMAI, U42, Sheet 3
0C	MACH*	Memory Address Count High	SCB68430 DMAI, U42, Sheet 3
0D	MACMH	Memory Address Count Mid-High	SCB68430 DMAI, U42, Sheet 3
0E	MACML	Memory Address Count Mid-Low	SCB68430 DMAI, U42, Sheet 3
0F	MACL	Memory Address Count Low	SCB68430 DMAI, U42, Sheet 3
25	DIVR	DMA Interrupt Vector Register	SCB68430 DMAI, U42, Sheet 3
27	IVR***	Interrupt Vector Register	SCB68430 DMAI, U42, Sheet 3
2D	CPR*	Channel Priority Register	SCB68430 DMAI, U42, Sheet 3
41	CR0**	Channel Register 0	MC68153 BIM, U57, Sheet 12
43	EICR (CR1)	Error Interrupt Control Register	MC68153 BIM, U57, Sheet 12
45	AICR (CR2)	Attention Interrupt Control Register	MC68153 BIM, U57, Sheet 12
47	DICR (CR3)	DMA Interrupt Control Register	MC68153 BIM, U57, Sheet 12
49	VR0**	Vector Register 0	MC68153 BIM, U57, Sheet 12
4B	EIVR (VR1)	Error Interrupt Vector Register	MC68153 BIM, U57, Sheet 12
4D	AIVR (VR2)	Attention Interrupt Vector Register	MC68153 BIM, U57, Sheet 12
4F	VR3**	Vector Register 3	MC68153 BIM, U57, Sheet 12
60	BCRH	Board Control Register High	U11, Sheet 25
61	BCRL	Board Control Register Low	U10 & U17, Sheet 25
62	IODRH	I/O Data Register High (16-Bit)	U18, Sheet 26, & U16, Sheet 28
63	IODRL	I/O Data Register Low (16-Bit)	U19, Sheet 26, & U15, Sheet 28
64	EMAR	Extended Memory Address Register	U65, Sheet 11
65	AMR	Address Modifier Register	U43, Sheet 11
66	MCR	Master Control Register	U32 & U56, Sheet 24
69	BSR	Board Status Register	U34, Sheet 24
6A	SCRQ	Set Cycle Request Command	U24, Sheet 10
6B			
6C	LDIDR	Load Input Data Register	U41, Sheet 21
6D			
70	ODR	Output Data Register (32-Bit)	U18, U19, Sheet 26 & U25, U26, Sheet 27
78	IDR	Input Data Register (32-Bit)	U21, U22, Sheet 29
7A			U15, U16, Sheet 28

MDR11W-A/T4.1-1

*Included for software compatibility.

**Register is not used.

***The IVR has two addresses for software compatibility (see SCB68430 Spec Sheet).

Table 4.2-1. DMA Interface (SCB68430) Register Bit Formats

CHANNEL STATUS REGISTER ($$XX00)** ($XXXX00)*** (READ/WRITE)

	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
CSR	CHANNEL OPERATION COMPLETE	NOT USED	NORMAL DEVICE TERMINATE	ERROR	CHANNEL ACTIVE	NOT USED	NOT USED	READY INPUT STATE
	0=NO 1=YES	(0)	(1)	0=NO 1=YES	0=NO 1=YES	(0)	(0)	0=0 V 1=3.5 V*

*Initialized to logic "one" at power-up. **Short I/O Address. ***Standard Address.

CHANNEL ERROR REGISTER ($XX01)* ($XXXX01)** (READ)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CER	NOT USED	NOT USED	NOT USED	ERROR CODE				
	(0)	(0)	(0)	0000=NO ERROR 01001=BUS ERROR 10001=SOFTWARE ABORT				

*Short I/O Address. **Standard Address.

DEVICE CONTROL REGISTER ($XX04)** ($XXXX04)*** (READ/WRITE)

	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
DCR	EXTERNAL REQUEST MODE	NOT USED	NOT USED	NOT USED	DATA SIZE	NOT USED	NOT USED	NOT USED
	0=BURST 1=CYCLE STEAL	(0)	(1)	(1)	*	(0)	(0)	(0)

*Should be programmed as a "zero" if the operand size (see Operation Control Register (OCR), bits 4 and 5) is BYTE, otherwise should be programmed as a "one". **Short I/O Address. ***Standard Address.

OPERATION CONTROL REGISTER ($XX05)** ($XXXX05)*** (READ/WRITE)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OCR	DIRECTION	NOT USED	OPERAND SIZE		NOT USED	NOT USED	NOT USED	NOT USED
	0=MEM TO DEV 1=DEV TO MEM	(0)	00=BYTE 01=WORD 10=LONGWORD* 11=32-BIT WORD		(0)	(0)	(1)	(0)

*Not applicable to the VMIVME-DR11W-A. **Short I/O Address. ***Standard Address.

MDR11W-A/T4.2-1

Table 4.2-1. DMA Interface (SCB68430) Register Bit Formats (Continued)

SEQUENCE CONTROL REGISTER ($XX06)** ($XXXX06)*** (READ/WRITE)

	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
SCR*	NOT USED*							
	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

*Dummy register. **Short I/O Address. ***Standard Address.

CHANNEL CONTROL REGISTER ($XX07)* ($XXXX07)** (READ/WRITE)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CCR	START	NOT USED	NOT USED	SOFTWARE ABORT	INTERRUPT ENABLE	NOT USED	NOT USED	NOT USED
	0=NO 1=YES	(0)	(0)	0=NO 1=YES	0=NO 1=YES	(0)	(0)	(0)

*Short I/O Address. **Standard Address.

MEMORY TRANSFER COUNT HIGH ($XX0A)* ($XXXX0A)** (READ/WRITE)

	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MTCH	MEMORY TRANSFER COUNT MSB							
	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8

*Short I/O Address. **Standard Address.

MEMORY TRANSFER COUNT LOW ($XX0B)* ($XXXX0B)** (READ/WRITE)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
MTCL	MEMORY TRANSFER COUNT LSB							
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

*Short I/O Address. **Standard Address.

MDR11W-A/T4.2-1/2

Table 4.2-1. DMA Interface (SCB68430) Register Bit Formats (Continued)

MEMORY ADDRESS COUNTER HIGH ($XX0C)** ($XXXX0C)*** (READ/WRITE)

	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MACH	NOT USED*							
	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

*Dummy register. **Short I/O Address. ***Standard Address.

MEMORY ADDRESS COUNTER MID-HIGH ($XX0D)* ($XXXX0D)** (READ/WRITE)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
MACMH	MEMORY ADDRESS COUNTER							
	BIT 23	BIT 22	BIT 21	BIT 20	BIT 19	BIT 18	BIT 17	BIT 16

*Short I/O Address. **Standard Address.

MEMORY ADDRESS COUNTER MID-LOW ($XX0E)* ($XXXX0E)** (READ/WRITE)

	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MACML	MEMORY ADDRESS COUNTER							
	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8

*Short I/O Address. **Standard Address.

MEMORY ADDRESS COUNTER LOW ($XX0F)* ($XXXX0F)** (READ/WRITE)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
MACL	MEMORY ADDRESS COUNTER							
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

*Short I/O Address. **Standard Address.

MDR11W-A/T4.2-1/3

Table 4.2-1. DMA Interface (SCB68430) Register Bit Formats (Concluded)

DMA INTERRUPT VECTOR REGISTER ($XX25, $XX27)** ($XXXX25, $XXXX27)*** (READ/WRITE)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DIVR	DONE INTERRUPT VECTOR							
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 *

*This bit is automatically set if an error occurred. This register is mapped to two locations to provide compatibility with other controllers. **Short I/O Address. ***Standard Address.

CHANNEL PRIORITY REGISTER ($XX2D)** ($XXXX2D)*** (READ/WRITE)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CPR	NOT USED*							
	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

*Dummy register. **Short I/O Address. ***Standard Address.

MDR11W-A/T4.2-1/4

BIT 14, BIT 13 - NOT USED.

BIT 12 - ERROR (See Notice). This bit will be set if the DMA operation was terminated due to an error condition.

BIT 11 - CHANNEL ACTIVE. This bit indicates that a DMA operation is in progress. It is automatically cleared upon termination of the operation.

BITS 10, 9 - NOT USED.

BIT 8 - READY INPUT. This bit indicates the state of the RDY input signal. A logic "zero" indicates an asserted RDY input signifying that data is ready.

4.2.1.2 CER Bit Definitions

BITS 7, 6, 5 - NOT USED.

BITS 4, 3, 2, 1, 0 - ERROR CODE. These five bits indicate the error type when an error occurs. This register is cleared when the error bit (CSR BIT 12) is cleared.

ERROR CODE	FUNCTION
00000	No Error
01001	Bus Error
10001	Software Abort

4.2.2 Device Control Register (DCR)/Operation Control Register (OCR)

These two registers control the data direction, data size, and request mode of the DMA transfers.

4.2.2.1 DCR Bit Definitions

BIT 15 - EXTERNAL REQUEST MODE. This bit is set to perform single cycle transfers and is cleared to perform burst transfers.

BITS 14, 13, 12 - NOT USED.

NOTICE

IF SET, CSR BIT 15 AND CSR BIT 12 MUST BE CLEARED BEFORE ANOTHER DMA OPERATION CAN BE STARTED. THESE BITS MAY BE CLEARED BY WRITING A "ONE" TO EACH BIT THAT IS SET. THE FOLLOWING 68000 INSTRUCTION WILL ACCOMPLISH THIS WITHOUT TESTING EACH OF THE BITS. *THE OTHER BITS IN THE CSR ARE UNAFFECTED BY A WRITE.*

```
MOVE. B        CSR,CSR
```

BIT 11 - DATA SIZE. This bit is read and written as the Logical "OR" of bits 4 and 5 in the Operation Control Register (OCR).

BITS 10, 9, 8 - NOT USED.

4.2.2.2 OCR Bit Definitions

BIT 7 - DIRECTION. This bit contains the DMA transfer direction.

0 =	Read from Memory Write to External Device
1 =	Write to Memory Read from External Device

BIT 6 - NOT USED.

BITS 5, 4 - OPERAND SIZE. These bits control the operand size.

BIT 5	BIT 4	OPERAND SIZE
0	0	BYTE
0	1	WORD
1	0	DUAL 16-BIT*
1	1	LONGWORD (32-BIT PER TRANSFER)

*The VMIVME-DR11W-A does not support dual 16-bit operations.

BITS 3, 2, 1, 0 - NOT USED.

4.2.3 Sequence Control Register (SCR)

The SCR is a dummy register provided for compatibility with other DMA controllers.

BITS 15 THROUGH 8 - NOT USED (Dummy Register).

4.2.4 Channel Control Register (CCR)

BIT 7 - START. This bit causes the DMA controller to start its operation. This bit must be set before the GO bit is set in the Board Control Register (BCR).

BITS 6, 5 - NOT USED.

BIT 4 - SOFTWARE ABORT. This bit allows current DMA operation to be aborted under software control. It sets the error bit in the CSR and the error code in the CER. The software abort bit must be cleared in order to clear the error bit in the CSR and the error code in the CER (refer to the following Software Abort Procedure).

Software Abort Procedure:

To abort an in-process DMA, perform the following steps:

STEP 1: Set bit 4 of the Channel Control Register (CCR, $0007) to a logic "one". Maintain bits 3 and 7 according to the desired control state.

STEP 2: Read the Channel Status Register (CSR, $0000) until bit 15 (operation complete) and bit 12 (error) are both equal to a logic "one", indicating a DMA complete with error.

STEP 3: Read the Channel Error Register to obtain the error condition for the DMA complete with error. Reset the software abort bit (bit 4), (CCR, $0007) to "zero".

NOTE:

THE CHANNEL ERROR REGISTER MUST BE READ WHILE THE ABORT BIT (BIT 4, CCR, $0007) IS ACTIVE (SET) OR THE CHANNEL ERROR REGISTER WILL INDICATE THE DEFAULT ERROR STATUS (BUS ERROR).

STEP 4: Perform normal DMA termination processing clean-up, resetting all of the CSR status bits to logic "zero" (CSR bit reset requires writing a logic "one" to each bit which is set to a logic "one".)

NOTE:

USE OF BURST MODE PRECLUDES THE USE OF SOFTWARE ABORT, SINCE THE BUS WILL BE RETAINED BY THE DR11W UNTIL THE DMA IS COMPLETED ONCE IT HAS BEEN INITIATED.

BIT 3 - INTERRUPT ENABLE. This bit enables the DMA interrupt request when a DMA operation is completed.

BITS 2, 1, 0 - NOT USED.

4.2.5 Memory Transfer Count (MTC) Registers

The two MTC registers, Memory Transfer Count High (MTCH) and Memory Transfer Count Low (MTCL), hold the number of desired transfers for the current operation. The transfer count can be set to FFFF (64 Ktransfers-1). The

"DONE" (Last TRANSFER Flag) signal is used to terminate the transfers in this case. However, setting the transfer counter to FFFF does cause an extra memory cycle to be requested beyond the actual word count (only if a read from memory). This should not be a problem unless it happens to be at the end of memory, in which case a bus error trap will be generated.

4.2.6 Memory Address Counter (MAC) Registers

The four MAC registers, Memory Address Count High (MACH), Memory Address Count Mid-High (MACMH), Memory Address Count Mid-Low (MACML), and Memory Address Count Low (MACL), hold the memory address for the DMA operation. The MACH Register is a dummy register and is provided for compatibility with other DMA set controllers. The other three implemented registers provide a 24-bit memory address for the DMA transfers. These registers should be loaded with the starting memory address before an operation is started. These registers are incremented during operation and may be read while an operation is in progress.

4.2.7 DMA Interrupt Vector Register (DIVR)

This register is used to store the interrupt vector used for the DMA complete (done) interrupt. This register is resident inside the SCB68430. It is mapped to two locations to provide compatibility with other DMA controllers. Note: Bit 0 is automatically set when an error occurs, and cannot be written. Thus, a DMA interrupt vector for normal conditions must be even and the error vector is automatically the next highest odd vector. Note that although the SCB68430 supplies the interrupt vector, the MC68153 provides the interrupt handshake; therefore, the MC68153 must be programmed to handle the external vector from the SCB68430 (i.e., bit 5 of the SCB68153's DMA Interrupt Control Register (DICR) must be set to a logic "one").

4.2.8 Channel Priority Register (CPR)

This register is a dummy register provided for compatibility with other DMA controllers.

4.3 PROGRAMMING THE MC68153 BIM

The MC68153 contains eight read/write registers. These registers include four Interrupt Control Registers (ICRs) that govern BIM operation and four Interrupt Vector Registers (IVRs) in which vector data are stored for use during an interrupt cycle. Each interrupt source is assigned a Control Register (CR0 to CR3) and a Vector Register (VR0 to VR3). The BIM Control Register assignments for the VMIVME-DR11W-A are as follows:

BIM CR	CONTROL REGISTER ASSIGNMENT
0	Not Used
1	Error Interrupt Control Register (EICR)
2	Attention Interrupt Control Register (AICR)
3	DMA Interrupt Control Register (DICR)

The BIM Vector Register assignments are as follows. The register that would normally be assigned to be the DMA Interrupt Vector Register, VR3, is not used since the Interrupt Vector for the DMA complete (done) interrupt is supplied by the SCB68430 DMAI's DMA Interrupt Vector Register as described in Section 4.2.

BIM VR	VECTOR REGISTER ASSIGNMENT
0	Not Used
1	Error Interrupt Vector Register (EIVR)
2	Attention Interrupt Vector Register (AIVR)
3	Not used

The register formats for these BIM CRs and VRs are detailed in Table 4.3-1, and the register bit descriptions are in the following paragraphs. Refer to the MC68153 data sheet contained in Appendix B for additional detailed operation and programming information for the BIM.

4.3.1 Error Interrupt Control Register (EICR)
Attention Interrupt Control Register (AICR)
DMA Interrupt Control Register (DICR)

Each of the registers controls an interrupt source. The three registers are identical and are each divided into six fields as follows:

BITS 7 AND 6 - FLAG CONTROL - Not Used

BIT 5 - VECTOR LOCATION. This bit should be cleared so that the MC68153 may use its internal vector register for an interrupt acknowledge and set to use an external interrupt vector. For the VMIVME-DR11W-A, this bit should be cleared in all control registers except for the DMA Interrupt Control Register (DICR); it should be set in the DICR because an external vector is generated by the SCB68430 DMA Controller Vector Register (DIVR).

Table 4.3-1. Bus Interrupt Module (MC68153) Register Bit Formats

ERROR INTERRUPT CONTROL REGISTER ($XX43)* ($XXXX43)** (READ/WRITE)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
EICR	FLAG BIT	FLAG BIT AUTO CLEAR	VECTOR 0=INT 1=EXT	INTERRUPT ENABLE	INTERRUPT AUTO CLEAR	IRQ LEVEL 2	IRQ LEVEL 1	IRQ LEVEL 0

*Short I/O Address. **Standard Address.

ATTENTION INTERRUPT CONTROL REGISTER ($XX45)* ($XXXX45)** (READ/WRITE)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
AICR	FLAG BIT	FLAG BIT AUTO CLEAR	VECTOR 0=INT 1=EXT	INTERRUPT ENABLE	INTERRUPT AUTO CLEAR	IRQ LEVEL 2	IRQ LEVEL 1	IRQ LEVEL 0

*Short I/O Address. **Standard Address.

DMA INTERRUPT CONTROL REGISTER ($XX47)* ($XXXX47)** (READ/WRITE)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DICR	FLAG BIT	FLAG BIT AUTO CLEAR	VECTOR 0=INT 1=EXT	NTERRUPT ENABLE	INTERRUPT AUTO CLEAR	IRQ LEVEL 2	IRQ LEVEL 1	IRQ LEVEL 0

*Short I/O Address. **Standard Address.

ERROR INTERRUPT VECTOR REGISTER ($XX4B)* ($XXXX4B)** (READ/WRITE)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
EIVR	**INTERRUPT VECTOR**							
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

*Short I/O Address. **Standard Address.

ATTENTION INTERRUPT VECTOR REGISTER ($XX4D)* ($XXXX4D)** (READ/WRITE)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
AIVR	**ATTENTION INTERRUPT VECTOR**							
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

*Short I/O Address. **Standard Address.

MDR11W-A/T4.3-1

BIT 4 - INTERRUPT ENABLE. This bit should be set to enable the corresponding interrupt.

BIT 3 - INTERRUPT AUTO CLEAR. This bit automatically clears the interrupt enable bit and the interrupt request output whenever the interrupt is acknowledged. It must be set in all control registers to provide VMEbus compatible timing; therefore, the interrupt enable bit is to be set after each interrupt.

BITS 2, 1, 0 - IRQ LEVEL BITS. These bits determine the interrupt level at which an interrupt request is made.

LEVEL	BIT 2	BIT 1	BIT 0
7	1	1	1
6	1	1	0
5	1	0	1
4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1
DISABLED	0	0	0

4.3.2 Error Interrupt Vector Register (EIVR) Attention Interrupt Vector Register (AIVR)

These registers have identical formats, and should be programmed to contain the vectors used for the associated interrupts. Any 8-bit integer may be used as determined by the system vector address configuration.

BITS 7 TO 0 - VECTOR ADDRESS BITS. These bits determine the 8-bit vector address for each associated interrupt.

4.4 PROGRAMMING THE ON-BOARD REGISTERS

There are eight additional sets of on-board registers listed in Table 4.1-1. Also included in the table are two special commands that support BIT (Built-in-Test) functions. The VMIVME-DR11W-A "on-board" register group consists of the Board Control Register (BCR), the VMIVME-DR11W compatible 16-bit I/O Data Register, the Extended Memory Address Register (EMAR), the Address Modifier Register (AMR), the Master Control Register (MCR), the Board Status Register (BSR), the 32-bit Output Data Register (ODR), and the 32-bit Input Data Register (IDR). The two special commands are the Load Input Data Register (LDIDR) and the Set Cycle Request (SCRQ) commands. Table 4.4-1 defines the On-Board Register bit formats for each of these registers. The On-Board Register set and the two special commands are further described in the following paragraphs.

Table 4.4-1. On-Board Register Bit Formats

BOARD CONTROL REGISTER ($XX60)* ($XXXX60)** (READ/WRITE)

BOARD CONTROL REGISTER HI ($XX60)* ($XXXX60)**

	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
BCRH	STATUS C READ ONLY	STATUS B READ ONLY	STATUS A READ ONLY	BURST REQUEST READ ONLY	C1 READ ONLY	ATTN READ ONLY	CLR GO BIT WRITE ONLY	MASTER RESET WRITE ONLY

BOARD CONTROL REGISTER LO ($XX61)* ($XXXX61)** - COMMAND***

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
BCRL	FNCT 3	FNCT 2	FNCT 1	RESERVE	LED CONTROL 0 = ON 1 = OFF	ENABLE ATTN INTERRUPT	CYCLE	GO

***Do not use a read-modify-write instruction (e.g., bit clear) when writing to the Board Control Register.

DR11W-A 16-BIT INPUT/OUTPUT DATA REGISTER ($XX62)* ($XXXX62)** (WORD READ/WRITE) WITH NO BYTE-SWAP SET***

I/O DATA REGISTER HI ($XX62)* ($XXXX62)**

	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
IODRH	I/O DATA							
	D15	D14	D13	D12	D11	D10	D9	D8

I/O DATA REGISTER LO ($XX63)* ($XXXX63)**

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
IODRL	I/O DATA							
	D7	D6	D5	D4	D3	D2	D1	D0

MDR11W-A/T4.4-1/1

*Short I/O Address

**Standard I/O Address

***Bit 6 (LONGWORD) of the AMR must be maintained to a logic "one" during 16-bit transfers utilizing the IODR.

Table 4.4-1. On-Board Register Bit Formats (Continued)

DR11W-A 16-BIT INPUT/OUTPUT DATA REGISTER ($XX62)* ($XXXX62)** AS READ WITH BYTE-SWAP SET***

I/O DATA REGISTER HI ($XX62)* ($XXXX62)**

	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
IODRH	I/O DATA							
	D7	D6	D5	D4	D3	D2	D1	D0

I/O DATA REGISTER LO ($XX63)* ($XXXX63)**

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
IODRL	I/O DATA							
	D15	D14	D13	D12	D11	D10	D9	D8

EXTENDED MEMORY ADDRESS REGISTER ($XX64)* ($XXXX64)** (READ/WRITE)

	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
EMAR	EXTENDED MEMORY ADDRESS (VMEbus A31 THROUGH A24 DURING DMA TRANSFERS)							
	BIT 31	BIT 30	BIT 29	BIT 28	BIT 27	BIT 26	BIT 25	BIT 24

ADDRESS MODIFIER REGISTER ($XX65)* ($XXXX65)** (READ/WRITE)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
AMR	NOT USED		ADDRESS MODIFIER					
		LONGWORD ***	AM5	AM4	AM3	AM2	AM1	AM0

MDR11W-A/T4.4-1/2

*Short I/O Address
**Standard I/O Address
***Bit 6 (LONGWORD) of the AMR must be maintained to a logic "one" during 16-bit transfers utilizing the IODR.

Table 4.4-1. On-Board Register Bit Formats (Continued)

MASTER CONTROL REGISTER HI ($XX66)* ($XXXX66)** (WORD READ/WRITE)

	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MCRH	NOT USED	ENABLE PROGRM SET CYC REQ	NOT USED	NOT USED	NOT USED	NOT USED	ENABLE ERROR IRQ	ENABLE WDT

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
MCRL	ENABLE PROG LD IN DTA REG	DISABLE CLK & RD INPUT REG	SET TEST MODE IN	SET TEST MODE OUT	SET INPUT WORD SWAP	SET INPUT BYTE SWAP	SET OUTPUT WORD SWAP	SET OUTPUT BYTE SWAP

BOARD STATUS REGISTER ($XX69)* ($XXXX69)** (BYTE READ)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
BSR	0	0	0	OTHER SIDE RDY FOR XFER	SET OUT CYCLE REQ	INPUT DATA READY	BUS T.O. ERR	WDT T.O. ERR

SET CYCLE REQUEST (DUMMY REGISTER) ($XX6A)* ($XXXX6A)** (WRITE ONLY)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
SCRQ	X	X	X	X	X	X	X	X

MDR11W-A/T4.4-1/3

*Short I/O Address
**Standard I/O Address

Table 4.4-1. On-Board Register Bit Formats (Continued)

LOAD INPUT DATA REGISTER ($XX6C)* ($XXXX6C)** (WRITE ONLY)

WORD WRITE

	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
LDIDR	INPUT DATA REGISTER							
	D15	D14	D13	D12	D11	D10	D9	D8

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
INPUT DATA REGISTER							
D7	D6	D5	D4	D3	D2	D1	D0

MDR11W-A/T4.4-1/4

*Short I/O Address
**Standard I/O Address

Table 4.4-1. On-Board Register Bit Formats (Continued)

LOAD INPUT DATA REGISTER ($XX6C)* ($XXXX6C)** (WRITE ONLY) WITH WORD-SWAP NOT SET

FIRST LONGWORD WRITE LOADS IDR BITS 31 TO 16 WITH DATA BITS D31 TO D16

LDIDR

BIT 31	BIT 30	BIT 29	BIT 28	BIT 27	BIT 26	BIT 25	BIT 24
INPUT DATA REGISTER							
D31	D30	D29	D28	D27	D26	D25	D24

BIT 23	BIT 22	BIT 21	BIT 20	BIT 19	BIT 18	BIT 17	BIT 16
INPUT DATA REGISTER							
D23	D22	D21	D20	D19	D18	D17	D16

SECOND LONGWORD WRITE LOADS IDR BITS 15 TO 0 WITH DATA BITS D15 TO D0

LDIDR

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
INPUT DATA REGISTER							
D15	D14	D13	D12	D11	D10	D9	D8

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
INPUT DATA REGISTER							
D7	D6	D5	D4	D3	D2	D1	D0

MDR11W-A/T4.4-1/5

*Short I/O Address
**Standard I/O Address

Table 4.4-1. On-Board Register Bit Formats (Continued)

LOAD INPUT DATA REGISTER ($XX6C)* ($XXXX6C)** (WRITE ONLY) WITH WORD-SWAP SET

FIRST LONGWORD WRITE LOADS IDR BITS 15 TO 0 WITH DATA BITS D31 TO D16

	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
LDIDR	INPUT DATA REGISTER							
	D31	D30	D29	D28	D27	D26	D25	D24

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
	INPUT DATA REGISTER							
	D23	D22	D21	D20	D19	D18	D17	D16

SECOND LONGWORD WRITE LOADS IDR BITS 31 TO 16 WITH DATA BITS D15 TO D0

	BIT 31	BIT 30	BIT 29	BIT 28	BIT 27	BIT 26	BIT 25	BIT 24
LDIDR	INPUT DATA REGISTER							
	D15	D14	D13	D12	D11	D10	D9	D8

	BIT 23	BIT 22	BIT 21	BIT 20	BIT 19	BIT 18	BIT 17	BIT 16
	INPUT DATA REGISTER							
	D7	D6	D5	D4	D3	D2	D1	D0

MDR11W-A/T4.4-1/6

*Short I/O Address
**Standard I/O Address

Table 4.4-1. On-Board Register Bit Formats (Continued)

DR11W-A 32-BIT OUTPUT DATA REGISTER ($XX70)* ($XXXX70)**

LONGWORD ONLY (WRITE)

	BIT 31	BIT 30	BIT 29	BIT 28	BIT 27	BIT 26	BIT 25	BIT 24
ODR	OUTPUT DATA							
	D31	D30	D29	D28	D27	D26	D25	D24

BIT 23	BIT 22	BIT 21	BIT 20	BIT 19	BIT 18	BIT 17	BIT 16
OUTPUT DATA							
D23	D22	D21	D20	D19	D18	D17	D16

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
OUTPUT DATA							
D15	D14	D13	D12	D11	D10	D9	D8

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUTPUT DATA							
D7	D6	D5	D4	D3	D2	D1	D0

MDR11W-A/T4.4-1/7

*Short I/O Address
**Standard I/O Address

Table 4.4-1. On-Board Register Bit Formats (Continued)

32-BIT OUTPUT DATA REGISTER ($XX70)* ($XXXX70)** WITH NO WORD-SWAP SET, NO BYTE-SWAP SET

FIRST WORD OUTPUT TO CABLE

DO 15	DO 14	DO 13	DO 12	DO 11	DO 10	DO 9	DO 8
FROM OUTPUT DATA REGISTER							
D31	D30	D29	D28	D27	D26	D25	D24

DO 7	DO 6	DO 5	DO 4	DO 3	DO 2	DO 1	DO 0
FROM OUTPUT DATA REGISTER							
D23	D22	D21	D20	D19	D18	D17	D16

SECOND WORD OUTPUT TO CABLE

DO 15	DO 14	DO 13	DO 12	DO 11	DO 10	DO 9	DO 8
FROM OUTPUT DATA REGISTER							
D15	D14	D13	D12	D11	D10	D9	D8

DO 7	DO 6	DO 5	DO 4	DO 3	DO 2	DO 1	DO 0
FROM OUTPUT DATA REGISTER							
D7	D6	D5	D4	D3	D2	D1	D0

MDR11W-A/T4.4-1/8

*Short I/O Address
**Standard I/O Address

Table 4.4-1. On-Board Register Bit Formats (Continued)

32-BIT OUTPUT DATA REGISTER ($XX70)* ($XXXX70)** WITH NO WORD-SWAP SET, NO BYTE-SWAP SET

FIRST WORD OUTPUT TO CABLE

DO 15	DO 14	DO 13	DO 12	DO 11	DO 10	DO 9	DO 8
FROM OUTPUT DATA REGISTER							
D15	D14	D13	D12	D11	D10	D9	D8

DO 7	DO 6	DO 5	DO 4	DO 3	DO 2	DO 1	DO 0
FROM OUTPUT DATA REGISTER							
D7	D6	D5	D4	D3	D2	D1	D0

SECOND WORD OUTPUT TO CABLE

DO 15	DO 14	DO 13	DO 12	DO 11	DO 10	DO 9	DO 8
FROM OUTPUT DATA REGISTER							
D31	D30	D29	D28	D27	D26	D25	D24

DO 7	DO 6	DO 5	DO 4	DO 3	DO 2	DO 1	DO 0
FROM OUTPUT DATA REGISTER							
D23	D22	D21	D20	D19	D18	D17	D16

MDR11W-A/T4.4-1/9

*Short I/O Address
**Standard I/O Address

Table 4.4-1. On-Board Register Bit Formats (Continued)

32-BIT OUTPUT DATA REGISTER ($XX70)* ($XXXX70)** WITH NO WORD-SWAP SET, BYTE-SWAP SET

FIRST WORD OUTPUT TO CABLE

DO 15	DO 14	DO 13	DO 12	DO 11	DO 10	DO 9	DO 8
FROM OUTPUT DATA REGISTER							
D23	D22	D21	D20	D19	D18	D17	D16

DO 7	DO 6	DO 5	DO 4	DO 3	DO 2	DO 1	DO 0
FROM OUTPUT DATA REGISTER							
D31	D30	D29	D28	D27	D26	D25	D24

SECOND WORD OUTPUT TO CABLE

DO 15	DO 14	DO 13	DO 12	DO 11	DO 10	DO 9	DO 8
FROM OUTPUT DATA REGISTER							
D7	D6	D5	D4	D3	D2	D1	D0

DO 7	DO 6	DO 5	DO 4	DO 3	DO 2	DO 1	DO 0
FROM OUTPUT DATA REGISTER							
D15	D14	D13	D12	D11	D10	D9	D8

MDR11W-A/T4.4-1/10

*Short I/O Address
**Standard I/O Address

Table 4.4-1. On-Board Register Bit Formats (Continued)

32-BIT OUTPUT DATA REGISTER ($XX70)* ($XXXX70)** WITH WORD-SWAP SET, BYTE-SWAP SET

FIRST WORD OUTPUT TO CABLE

DO 15	DO 14	DO 13	DO 12	DO 11	DO 10	DO 9	DO 8
FROM OUTPUT DATA REGISTER							
D7	D6	D5	D4	D3	D2	D1	D0

DO 7	DO 6	DO 5	DO 4	DO 3	DO 2	DO 1	DO 0
FROM OUTPUT DATA REGISTER							
D15	D14	D13	D12	D11	D10	D9	D8

SECOND WORD OUTPUT TO CABLE

DO 15	DO 14	DO 13	DO 12	DO 11	DO 10	DO 9	DO 8
FROM OUTPUT DATA REGISTER							
D23	D22	D21	D20	D19	D18	D17	D16

DO 7	DO 6	DO 5	DO 4	DO 3	DO 2	DO 1	DO 0
FROM OUTPUT DATA REGISTER							
D31	D30	D29	D28	D27	D26	D25	D24

MDR11W-A/T4.4-1/11

*Short I/O Address
**Standard I/O Address

Table 4.4-1. On-Board Register Bit Formats (Continued)

DR11W-A 32-BIT INPUT DATA REGISTER ($XX78)* ($XXXX78)**

NO WORD-SWAP SET, NO BYTE-SWAP SET (LONGWORD READ ONLY)

	BIT 31	BIT 30	BIT 29	BIT 28	BIT 27	BIT 26	BIT 25	BIT 24
IDR	INPUT DATA WORD 1 (READ ONLY)							
	D15	D14	D13	D12	D11	D10	D9	D8

BIT 23	BIT 22	BIT 21	BIT 20	BIT 19	BIT 18	BIT 17	BIT 16
INPUT DATA WORD 1 (READ ONLY)							
D7	D6	D5	D4	D3	D2	D1	D0

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
INPUT DATA WORD 2 (READ ONLY)							
D15	D14	D13	D12	D11	D10	D9	D8

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
INPUT DATA WORD 2 (READ ONLY)							
D7	D6	D5	D4	D3	D2	D1	D0

MDR11W-A/T4.4-1/12

*Short I/O Address
**Standard I/O Address

Table 4.4-1. On-Board Register Bit Formats (Continued)

DR11W-A 32-BIT INPUT DATA REGISTER ($XX78)* ($XXXX78)**

WORD-SWAP SET, BYTE-SWAP SET (LONGWORD READ ONLY)

	BIT 31	BIT 30	BIT 29	BIT 28	BIT 27	BIT 26	BIT 25	BIT 24
IDR	INPUT DATA WORD 2 (READ ONLY)							
	D15	D14	D13	D12	D11	D10	D9	D8

BIT 23	BIT 22	BIT 21	BIT 20	BIT 19	BIT 18	BIT 17	BIT 16
INPUT DATA WORD 2 (READ ONLY)							
D7	D6	D5	D4	D3	D2	D1	D0

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
INPUT DATA WORD 1 (READ ONLY)							
D15	D14	D13	D12	D11	D10	D9	D8

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
INPUT DATA WORD 1 (READ ONLY)							
D7	D6	D5	D4	D3	D2	D1	D0

MDR11W-A/T4.4-1/13

*Short I/O Address
**Standard I/O Address

Table 4.4-1. On-Board Register Bit Formats (Continued)

DR11W-A 32-BIT INPUT DATA REGISTER ($XX78)* ($XXXX78)**

NO WORD-SWAP SET, BYTE-SWAP SET (LONGWORD READ ONLY)

	BIT 31	BIT 30	BIT 29	BIT 28	BIT 27	BIT 26	BIT 25	BIT 24
IDR	INPUT DATA WORD 1 (READ ONLY)							
	D7	D6	D5	D4	D3	D2	D1	D0

BIT 23	BIT 22	BIT 21	BIT 20	BIT 19	BIT 18	BIT 17	BIT 16
INPUT DATA WORD 1 (READ ONLY)							
D15	D14	D13	D12	D11	D10	D9	D8

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
INPUT DATA WORD 2 (READ ONLY)							
D7	D6	D5	D4	D3	D2	D1	D0

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
INPUT DATA WORD 2 (READ ONLY)							
D15	D14	D13	D12	D11	D10	D9	D8

MDR11W-A/T4.4-1/14

*Short I/O Address
**Standard I/O Address

Table 4.4-1. On-Board Register Bit Formats (Concluded)

DR11W-A 32-BIT INPUT DATA REGISTER ($XX78)* ($XXXX78)**

WORD-SWAP SET, BYTE-SWAP SET (LONGWORD READ ONLY)

	BIT 31	BIT 30	BIT 29	BIT 28	BIT 27	BIT 26	BIT 25	BIT 24
IDR	INPUT DATA WORD 2 (READ ONLY)							
	D7	D6	D5	D4	D3	D2	D1	D0

BIT 23	BIT 22	BIT 21	BIT 20	BIT 19	BIT 18	BIT 17	BIT 16
INPUT DATA WORD 2 (READ ONLY)							
D15	D14	D13	D12	D11	D10	D9	D8

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
INPUT DATA WORD 1 (READ ONLY)							
D7	D6	D5	D4	D3	D2	D1	D0

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
INPUT DATA WORD 1 (READ ONLY)							
D15	D14	D13	D12	D11	D10	D9	D8

MDR11W-A/T4.4-1/15

*Short I/O Address
**Standard I/O Address

4.4.1 Board Control Register (BCR)

The BCR is a 16-bit Status and Control Register that is used to communicate with the external DR11W compatible device via the interface status and control signals over the interface cable. While the upper eight bits of the BCR, BCRH, are not affected, all eight of the lower BCR bits, BCRL, are reset to logic "zero" by the Master Reset Command.

4.4.1.1 BCRH (BCR High) Bit Descriptions

BIT 15 - STATUS C. This bit is user-defined. It indicates the state of the FNCT 1 bit of the other DR11W device in BACK-TO-BACK MODE.

STATUS C	OTHER DR11W
0	Transmitter
1	Receiver

BIT 14 - STATUS B. This bit is user-defined. It indicates the state of the other DR11WA's FNCT 2 bit in BACK-TO-BACK MODE.

BIT 13 - STATUS A. This bit is user-defined. It indicates the state of the other DR11W's FNCT 3 bit in BACK-TO-BACK MODE.

STATUS A	OTHER DR11W
0	Burst Transfers
1	Normal (Single Cycle) Transfers

BIT 12 - BURST REQUEST. This bit indicates the state of the request line. In the BACK-TO-BACK MODE, it is the same as STATUS A.

BIT 11 - C1. This bit indicates the state of the C1 line. In the BACK-TO-BACK MODE, C1 is controlled by FUNCTION 1 of the same board.

C1	PROGRAM OCR BIT 5	THIS DR11W
0	0	Transmitter
1	1	Receiver

BIT 10 - ATTN. This bit indicates the state of the ATTN line. In BACK-TO-BACK MODE, it is the same as Status B. A positive edge on the ATTN input will generate an interrupt (if enabled); therefore, the other DR11W can generate an interrupt by pulsing the ATTN line.

BIT 9 - CLR GO BIT (CLEAR GO BIT). When set to a logic "one", this bit clears the state of the GO BIT to a logic "zero". This write only bit is not stored and does not need to be reset to "zero".

BIT 8 - MASTER RESET. When set to a logic "one", this bit clears all resettable registers and status on the VMIVME-DR11W-A interface as follows. This bit is a write-only control bit that is not stored, and does not need to be reset to "zero".

Board Control Register - Bits 15 through 10, Not Affected; 9, Momentarily Asserted; 8, This Bit Momentarily Asserted; 7 Through 0, Cleared.

I/O Data Register High/Low - Not Affected
Extended Memory Address Register - All Bits
Address Modifier Register - All Bits
Master Control Register - All Bits
Board Status Register - All Bits
Output Data Register - Not Affected
Input Data Register - Not Affected

4.4.1.2 BCRL (BCR Low) Bit Descriptions

BIT 7 - FUNCTION 3. This bit is user-defined. It drives the STATUS A and BURST RQ lines of the other DR11W in BACK-TO-BACK MODE.

FNCT 1	TELLS OTHER DR11W TO USE
0	Burst Transfers
1	Normal (Single Cycle) Transfers

BIT 6 - FUNCTION 2. This bit is user-defined. It drives the STATUS B and ATTN lines of the other DR11W in BACK-TO-BACK MODE. It can be pulsed high to generate an attention interrupt.

BIT 5 - FUNCTION 1. This bit is defined. In the BACK-TO-BACK MODE, it drives its own C1 line and the STATUS C line of the other DR11W.

FNCT 1	THIS VMIVME-DR11W-A
0	Transmitter
1	Receiver

This bit should be the complement of STATUS C.

BIT 4 - RESERVED.

BIT 3 - LED CONTROL (SET FAIL LED LOW). When set to a logic "zero", this bit lights the red light-emitting diode (LED) located on the front panel of the VMIVME-DR11W-A interface board. It is usually used as a Fail LED to signify that the VMIVME-DR11W-A interface has in some way failed. When set to a logic "one", the LED is extinguished, indicating continuing successful operation as defined by the user.

0 = LED ON
1 = LED OFF

BIT 2 - ENABLE ATTN INTERRUPT. This bit is used to control the ATTN interrupt flip-flop. This bit must be set in addition to AICR bit 4 to enable external attention interrupts.

BIT 1 - CYCLE. This bit is used to start the first handshake cycle when the VMIVME-DR11W-A is the transmitter. It may be set at the same time as the GO bit.

BIT 0 - GO. This bit sets up the handshake logic to perform transfers and causes a pulse to be generated on the VMIVME-DR11W-A's GO output. It must be set before the start of each block of transfers. The DMA start bit (bit 7 of CCR) must be set before the GO bit is set.

4.4.2 VMIVME-DR11W-A I/O Data Register (IODR) High/Low*

The 16-bit Input/Output Data Register (IODR) is provided to maintain data register compatibility with the 16-bit I/O Data Register of the VMIVME-DR11W, VMEbus to DR11W Interface board. This register is written by the CPU either when operating as bus master (DMA transfer) or under program control. The outputs of the Output Data Register (ODR) are output 16-bits at a time via high-speed bus drivers toward the external DR11W (or VMIVME-DR11W-A compatible) interface during an output transfer. The IODR must be written at this address ($XX62) as a 16-bit word only.

* Bit 6 (LONGWORD) of the Address Modifier Register (AMR) must be maintained to a logic "one" during 16-bit transfers.

The 16-bit IODR may be read under DMA control or program control. It is capable of 16-bit word transfers only. The Input Data Register receives and stores the data that are transferred from the external DR11W (or VMIVME-DR11W-A compatible) interface during an input transfer.

BITS 15 THROUGH 0 - I/O DATA D15 THROUGH D8. (Input/Output Data bits D15 through D0).

4.4.3 Extended Memory Address Register (EMAR)

The EMAR, a read/write register, holds bits A24 through A31 of VMEbus addresses for DMA transfers. It is resettable under program control using the Master Reset command or at power-up by the SYSRESET VMEbus signal. It must be loaded when the Address Modifier Register selects an extended address. The contents of this register may be examined with a read at any time under program control by the VMEbus CPU.

BITS 15 THROUGH 8 - EMA BIT 31 THROUGH EMA BIT 24.

EMA bits 31 through 24 correspond to VMEbus address bits A31 through A24.

4.4.4 Address Modifier Register (AMR)*

The AMR is a read/write register that stores the address modifier and longword bits that are asserted while the VMIVME-DR11W-A is bus master. It will be reset by the assertion of the DR11W-A Master Reset command under program control or by the VMEbus signal SYSRESET at power-up.

BIT 7 - Not Used.

BIT 6 - Longword. This bit controls the assertion of the VMEbus LWORDL signal and selects either 8-, 16-, or 32-bit VMEbus transfers. The operand size selected in the OCR bits 4 and 5 (in the SCB68430 at address $XX05) must be programmed for the same transfer size as shown in the following table:

LONGWORD	TRANSFER SIZE	BITS 5, 4 OF OCR
0	32-Bit	11
1	16-Bit	01
1	8-Bit	00

BITS 5, 4, 3, 2, 1, AND 0 - ADDRESS MODIFIER BITS. These bits drive the VMEbus Address Modifier Bits 5, 4, 3, 2, 1, and 0, respectively, during a DMA transfer. Refer to Table 4.4.4-1 for a description of the VMEbus address modifier codes.

* Bit 6 (LONGWORD) of the Address Modifier Register (AMR) must be maintained to a logic "one" during 16-bit transfers.

Table 4.4.4-1. Address Modifier Codes

HEXADECIMAL CODE	ADDRESS MODIFIER						FUNCTION	DEFINED BY
	5	4	3	2	1	0		
3F	H	H	H	H	H	H	Standard Supervisory Ascending Access	VMEbus Spec
3E	H	H	H	H	H	L	Standard Supervisory Program Access	VMEbus Spec
3D	H	H	H	H	L	H	Standard Supervisory Data Access	VMEbus Spec
3C	H	H	H	H	L	L	Undefined	Reserved
3B	H	H	H	L	H	H	Standard Nonprivileged Ascending Access	VMEbus Spec
3A	H	H	H	L	H	L	Standard Nonprivileged Program Access	VMEbus Spec
39	H	H	H	L	L	H	Standard Nonprivileged Data Access	VMEbus Spec
38	H	H	H	L	L	L	Undefined	Reserved
30-37	H	H	L	X	X	X	Undefined	Reserved
2F	H	L	H	H	H	H	Undefined	Reserved
2E	H	L	H	H	H	L	Undefined	Reserved
2D	H	L	H	H	L	H	Short Supervisory I/O Access	VMEbus Spec
2C	H	L	H	H	L	L	Undefined	Reserved
2B	H	L	H	L	H	H	Undefined	Reserved
2A	H	L	H	L	H	L	Undefined	Reserved
29	H	L	H	L	L	H	Short Nonprivileged Access	VEMbus Spec
28	H	L	H	L	L	L	Undefined	Reserved
20-27	H	L	L	X	X	X	Undefined	Reserved
10-1F	L	H	X	X	X	X	Undefined	User
0F	L	L	H	H	H	H	Extended Supervisory Ascending Access	VMEbus Spec
0E	L	L	H	H	H	L	Extended Supervisory Program Access	VMEbus Spec
0D	L	L	H	H	L	H	Extended Supervisory Data Access	VMEbus Spec
0C	L	L	H	H	L	L	Undefined	Reserved
0B	L	L	H	L	H	H	Extended Nonprivileged Ascending Access	VMEbus Spec
0A	L	L	H	L	H	L	Extended Nonprivileged Program Access	VMEbus Spec
09	L	L	H	L	L	H	Extended Nonprivileged Data Access	VMEbus Spec
08	L	L	H	L	L	L	Undefined	Reserved
00-07	L	L	L	X	X	X	Undefined	Reserved

MDR11W-A/T4.4.4-1

4.4.5 Master Control Register (MCR)

The MCR is a 16-bit register that is used to control the transfer activities of the VMIVME-DR11W-A. All bits of the MCR are cleared to a logic "zero" by the assertion of the VMEbus SYS RESET* signal or by the assertion of the Master Reset command via the Board Control Register (BCR).

BIT 15 - Not Used.

BIT 14 - ENABLE PROGRM SET CYC REQ (Enable Program Set Cycle Request). This bit, if set to a logic "one", enables the program to use the SET CYCLE REQUEST command to assert the Send Next Output Flip-Flop, requesting the next 16-bit output data word (see also paragraph 4.4.8). As long as this bit remains asserted, the Cycle Req A and the Cycle Req B input interface signals will be inhibited from setting the Send Next Output Flip-Flop.

BITS 13, 12, 11, 10 - Not Used.

BIT 9 - ENABLE ERROR IRQ (Enable Error Interrupt Request). This bit, if set to a logic "one", enables the Error Interrupt Request to be set by the assertion of one of the three following error conditions:

Cable WDT Time-out Error (18.3 ms)
Bus Time-out Error (18.3 ms)
Bus Error

BIT 8 - ENABLE WDT (Enable Watchdog Timer). This bit, if set, will enable the Watchdog Timer (18.3 ms) to generate either a Cable WDT Time-out Error and an Error Interrupt Request (if enabled), or:

(a) if the GO BIT is set, and if the VMIVME-DR11W-A is receiving input data from the attached DR11W compatible host interface and, the input data word is not received in the Input Data Register within the Watchdog Time-out period (18.3 ms) from the assertion of the GO BIT, for the first DMA input data transfer request, or from the receipt and storage of the previous input data word, for subsequent DMA input data transfers, or

(b) if the GO BIT is set and the VMIVME-DR11W-A is outputting data and the next DMA output transfer is not requested by the VMIVME-DR11W-A interface "Cycle" flip-flop and received in the Output Data Register within the Watchdog Time-out Period (18.3 ms) from the previous output cycle, or

(c) a Bus Time-out Error and an Error Interrupt Request (if enabled), if, while the VMIVME-DR11W-A interface is acting as a VMEbus Master, the DTACK acknowledge signal is not received asserted from the addressed VMEbus slave within the Watchdog Timer Time-out Period (which begins at the assertion of the VMEbus DATA STROBE* signal by the VMIVME-DR11W-A Master).

BIT 7 - ENABLE PROG LD IN DTA REG (Enable Programmed Load Input Data Register). This bit, when set to a logic "one", enables the program to load the Input Data Register (IDR) by issuing Load Input Data Register commands under program control. Refer to the description of the LOAD INPUT DATA REGISTER COMMAND in paragraph 4.4.7.

BIT 6 - DISABLE CLK & RD INPUT REG (Disable Clock and Read Input Register). Assertion of this bit will disable the programmed I/O CLOCK AND READ INPUT REGISTER mode of operation.

The programmed I/O CLOCK AND READ INPUT REGISTER MODE is the normal mode of program controlled Input Data Register (IDR) read operation. In this mode, the IDR is strobed (clocked) during execution of the Read Input Data Register command and is, therefore, updated (loaded) with the current states of the DI15 through DI00 ribbon-cable interface signals as the read occurs. With the DISABLE CLK & RD INPUT REG bit asserted, the current contents of the Input Data Register (IDR) can be read without being updated to the current interface signal states. Thus, the last value loaded in the IDR by the programmed Load Input Data Register command or by the busy signal from the other VMIVME-DR11W-A compatible interface will not be lost when read under programmed control with the Read Input Data Register command.

With the DISABLE CLK & RD INPUT REG bit not asserted, a test loop can be established consisting of a single DMA output transfer cycle, followed by a Read Input Data Register command (with the GO, SET TEST MODE IN, and SET TEST MODE OUT bits **NOT** asserted) to verify single cycle DMA output operation. While only the interface cable DO15 through DO00 signals need to be wrapped back to the DI15 through DI00 signals for verification in this mode, a full wraparound cable, such as that described later in the BIT 5 description, may be used.

BIT 5 - SET TEST MODE IN. When set to a logic "one", this bit enables programmed control of VMIVME-DR11W-A input functions that would otherwise be prevented from operating while the TRANSMIT MODE is enabled with the FUNCTION 1 bit set to a logic "zero", or while the other board's READY signal is set to a logic "one". This enables the VMIVME-DR11W-A to operate in a program controlled input (receive) mode for BIT, concurrent with DMA operation of the output data function.

Enabling the TEST MODE IN in the TRANSMIT MODE with the FUNCTION 1 bit set to a logic "zero" allows an Input Data Register strobe, whether initiated by a program generated LD IN DATA REQ command, a SET CYCLE REQ command or an externally generated CYCLE REQUEST A (or CYCLE REQUEST B) input signal, to set the Input Data Ready status as an indication that the Input Data Register has been loaded with a new data word (or longword, if /LWORD is asserted in the AMR).

This feature can be utilized to establish a test WRAPAROUND MODE. Using a special wraparound test cable, the user interface signals of connector P3 can be connected to those of connector P4. With this cable, the Data Output Bus, DO00 to DO15, connects to the Data Input Bus, DI15 to DI00, BUSY connects to CYCLE REQUEST A, etc., so that the interface signal connections are identical to those that would be established in a back-to-back VMIVME-DR11W-A connection. With the TEST MODE IN asserted and the cycle steal (16-bit word or 32-bit longword) DMA TRANSMIT mode established, the assertion of the Input Data Ready status bit of the BSR can be used to queue a read back of the transmitted data via the IDR under program control for data verification. Figure 4.4.5-1 (A and B) provides a hypothetical program functional flow diagram for an exemplary test routine utilizing the TEST MODE IN feature.

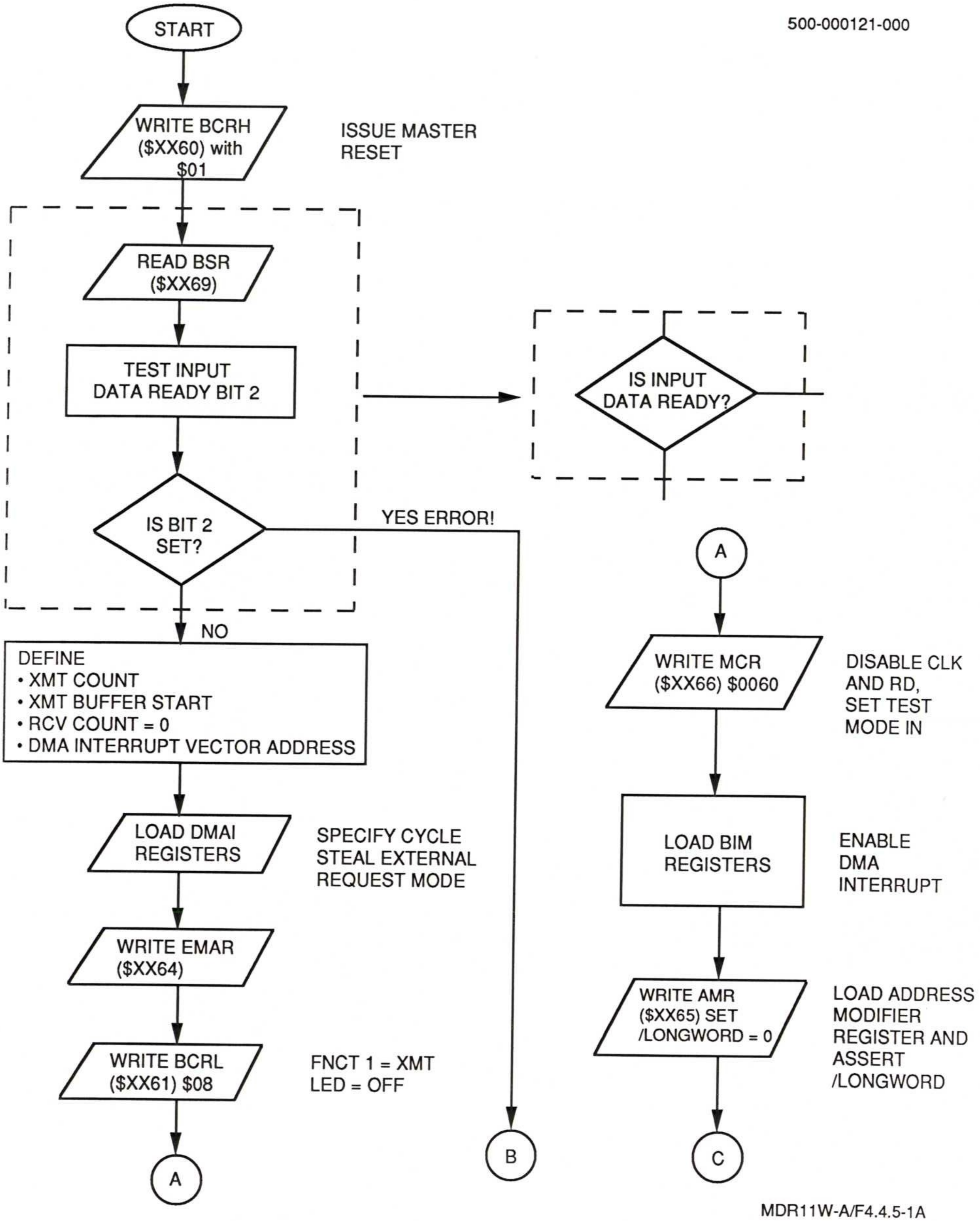
BIT 4 - SET TEST MODE OUT. When set to a logic "one", this bit enables programmed control of VMIVME-DR11W-A output functions that would otherwise be prevented from operating while the RECEIVE MODE is enabled with the FUNCTION 1 bit set to a logic "one", or by an active, unasserted other board READY signal. Thus, programmed control of these output functions is made possible concurrent with DMA operation of the Receive function for BIT.

An asserted SET TEST MODE OUT bit enables:

a. the BUSY handshake strobe to signify the presence of new output data on the interface cable during program controlled outputs, concurrent with an active DMA RECEIVE MODE (during which the GO and FUNCTION 1 bits are set),

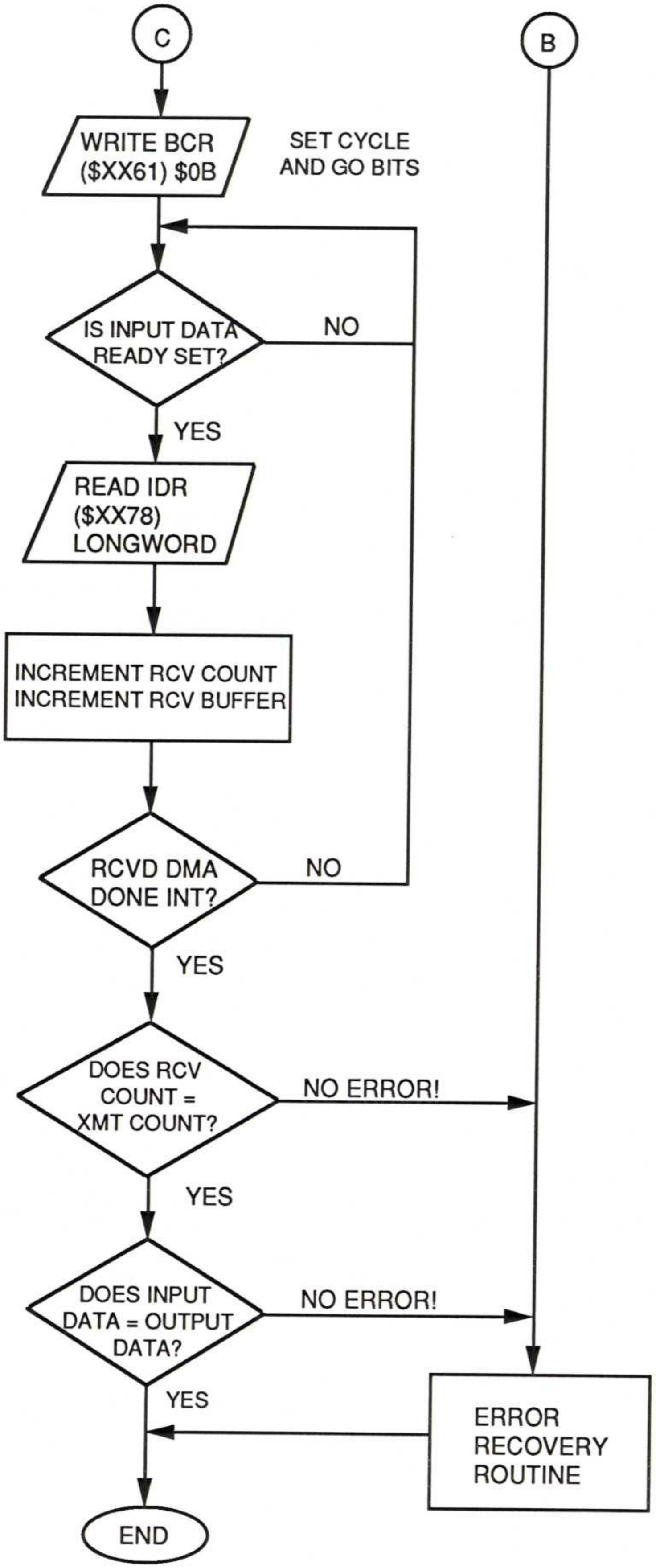
b. the Output Word Swap function during programmed output operation, concurrent with the DMA receive function (normally inhibited when the GO bit is asserted along with the FUNCTION 1 bit or with a deasserted OTHER SIDE READY FOR XFER H signal), or

c. the CYCLE REQ A signal, CYCLE REQ B signal or the SET CYCLE REQ command (user-selected) to set the Send Next Output request for more output data while the interface is operating in the DMA RECEIVE MODE (also normally inhibited when the GO bit is asserted with the FUNCTION 1 bit or with a deasserted OTHER SIDE READY FOR XFER H signal).



MDR11W-A/F4.4.5-1A

Figure 4.4.5-1A. DMA Transmit (XMT), Program Controlled Receive (RCV), Wraparound Test Mode Program Functional Flow Diagram, Longword



MDR11W-A/F4.4.5-1B

Figure 4.4.5-1B. DMA Transmit (XMT), Program Controlled Receive (RCV), Wraparound Test Mode, Program Functional Flow Diagram, Longword, (Continued)

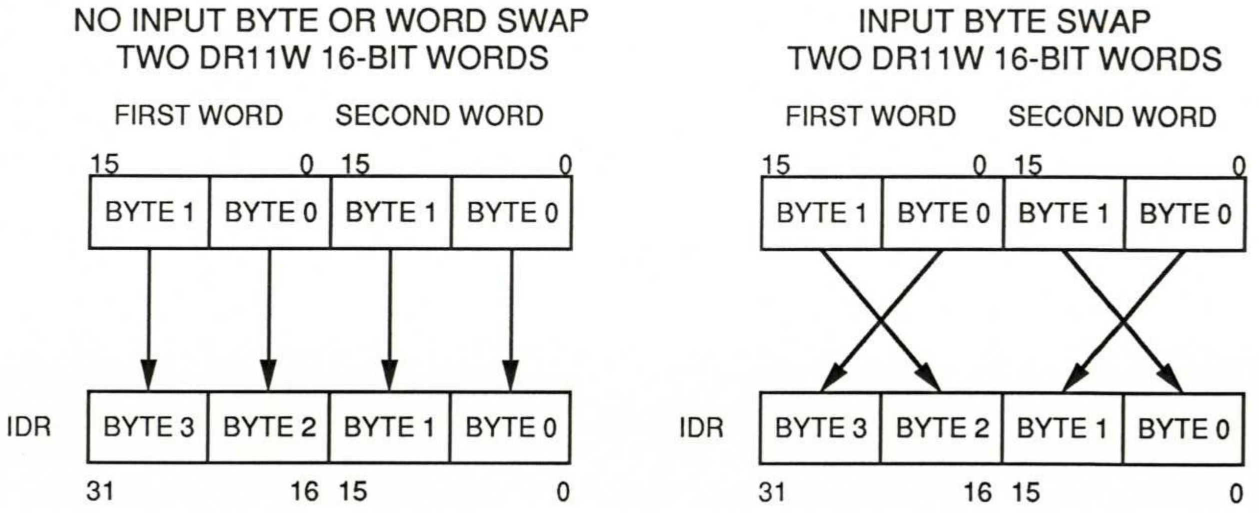
In a manner similar to that described for the TEST MODE IN bit previously, a WRAPAROUND test mode can be established that allows multiple program controlled output transmissions wrapped around from connector P3 to connector P4 with a WRAPAROUND cable to be received and input in a DMA Receive mode. To set up this mode, the SET TEST MODE OUT bit must be asserted, along with the DISABLE CLK & RD IN REG bit of the MCR.

The FUNCTION 1 bit of the BCR should also be asserted throughout the test to establish the receive mode. It will be necessary to specify a cycle steal mode when setting up the DMAI chip so that programmed controlled transfers can occur concurrent with the DMA receive process. When set up is completed, and GO has been asserted to allow DMA input transfers (Writes) to occur, the program should issue a Write to the ODR to start the data transfer process. Next, as Figure 4.4.5-2 illustrates, the SET OUT CYCLE REQ bit of the BSR should be tested to determine whether or not the DMA Receive has been completed. A transition from a logic "zero" to a "one" signifies that the data input cycle has been completed, and that the output circuit is prepared for the next program controlled Write to the ODR. While the transition may or may not be detectable, depending on other processor activities, a logic "one" signifies readiness for the next output data word. See Figure 4.4.5-2 for an example of a hypothetical test program functional flow diagram utilizing this feature in a 32-bit longword transfer mode.

BIT 3 - SET INPUT WORD SWAP. This bit allows control of the 16-bit word position within each 32-bit longword Input Data Register (IDR). When asserted, the input word swap function will become operational for 32-bit transfers, if the FUNCTION 1 control bit is set to receive (logic "one") and, if enabled, the other board READY signal has transitioned to READY active low. The set input word swap function can also be enabled for 32-bit transfers, concurrent with an active OUTPUT TRANSFER MODE with FUNCTION 1 set to logic "zero", irrespective of the state of the other board's READY signal, by placing the VMIVME-DR11W-A in the TEST MODE with the TEST MODE IN BIT asserted.

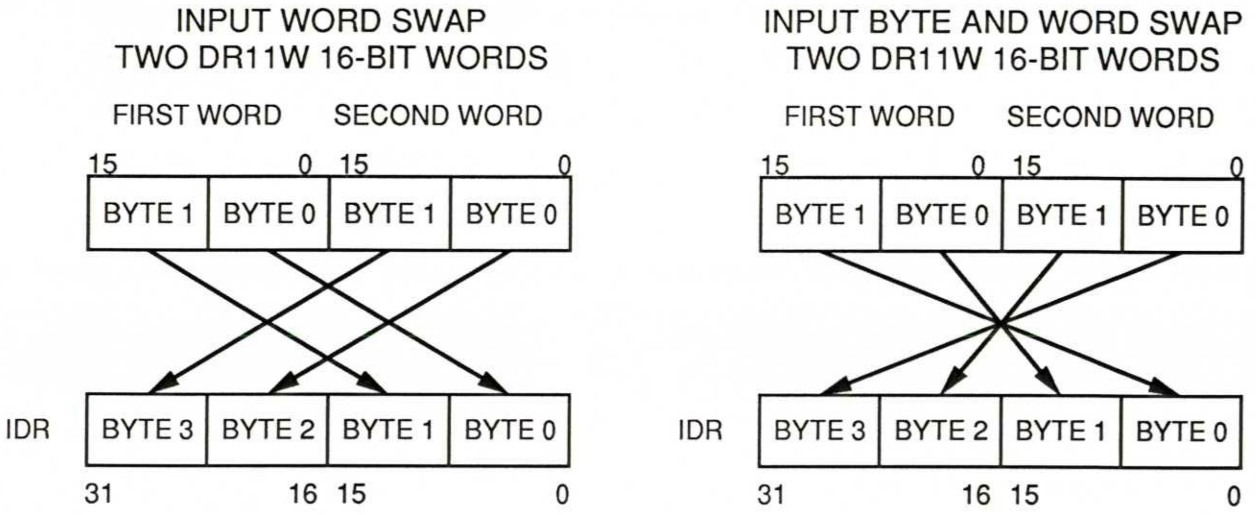
If set to a logic "zero", the first word received by the VMIVME-DR11W-A will be directed by the load pointer to be loaded into the High Word Input Data Register as IDR bits 31 through 16. The second word received will be loaded into the Low Word Input Data Register as IDR bits 15 through 0. If byte swapping has not been also specified, the bit order within each 16-bit word will be maintained. Refer to Figure 4.4.5-1 for an illustration of the input data bit organization with the SET INPUT WORD SWAP bit NOT set.

If set to a logic "one", the order of the words loaded into the Input Data Register (IDR) is reversed (or swapped). The first word received is loaded into the Low Word, while the second word received is loaded into the High Word, as directed by the load pointer. Figure 4.4.5-2 illustrates this word reversal. Again, unless byte swapping has been specified, the bit order within each 16-bit word is maintained as shown. If byte swapping has been also specified, the byte position within each 16-bit word will be swapped as described below. It is not possible to swap input words to the 16-bit IODR located at word address $XX62.



MDR11W-A/F4.4.5-1

Figure 4.4.5-1. Input Data Formats *WITHOUT* Word Swap



MDR11W-A/F4.4.5-2

Figure 4.4.5-2. Input Data Formats *WITH* Word Swap

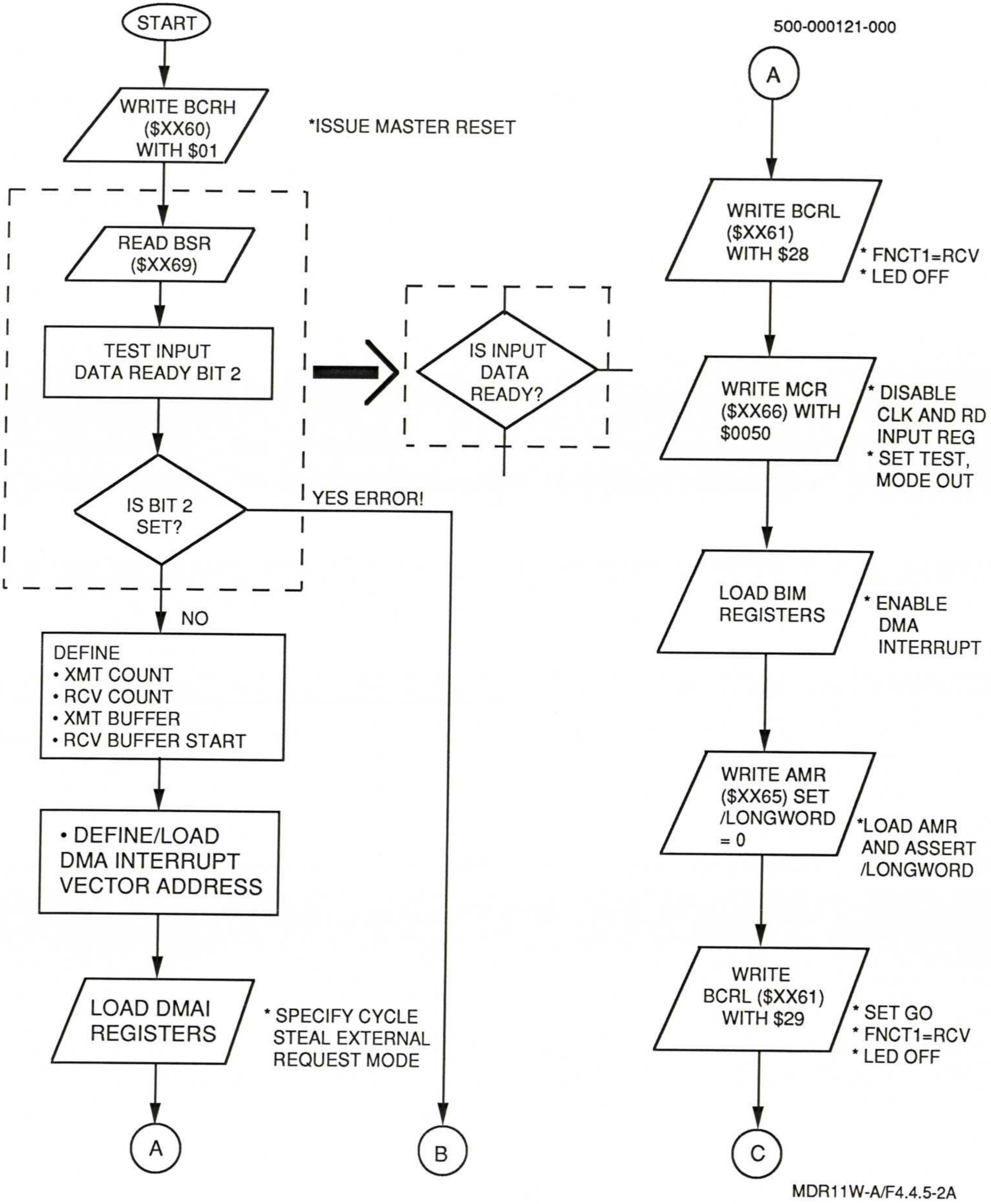
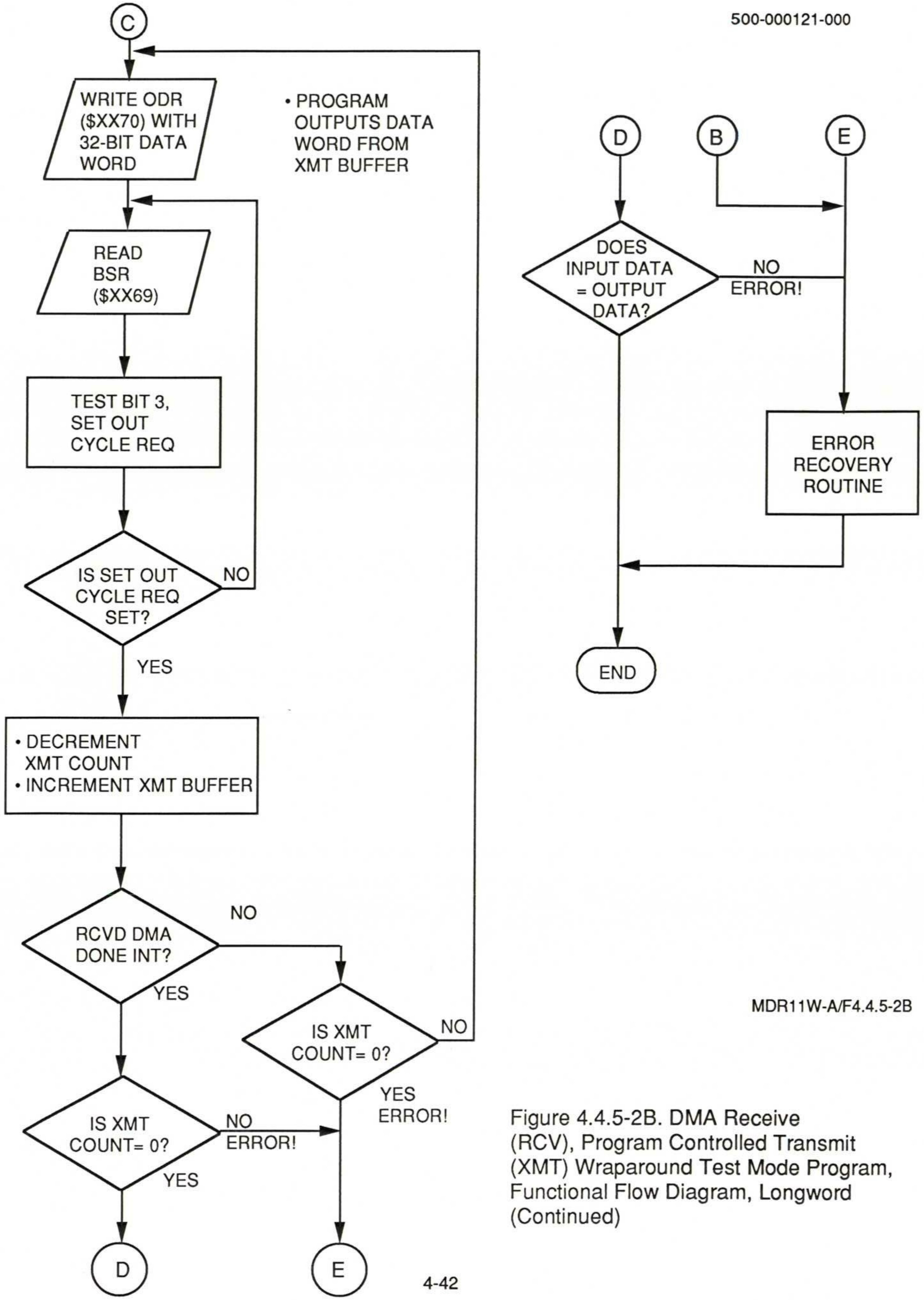

START
WRITE BCRH ($XX60) WITH $01
*ISSUE MASTER RESET
READ BSR ($XX69)
TEST INPUT DATA READY BIT 2
IS BIT 2 SET?
YES ERROR!
NO
IS INPUT DATA READY?
DEFINE
• XMT COUNT
• RCV COUNT
• XMT BUFFER
• RCV BUFFER START
• DEFINE/LOAD DMA INTERRUPT VECTOR ADDRESS
LOAD DMAI REGISTERS
* SPECIFY CYCLE STEAL EXTERNAL REQUEST MODE
A
B
A
WRITE BCRL ($XX61) WITH $28
* FNCT1=RCV
* LED OFF
WRITE MCR ($XX66) WITH $0050
* DISABLE CLK AND RD INPUT REG
* SET TEST, MODE OUT
LOAD BIM REGISTERS
* ENABLE DMA INTERRUPT
WRITE AMR ($XX65) SET /LONGWORD = 0
*LOAD AMR AND ASSERT /LONGWORD
WRITE BCRL ($XX61) WITH $29
* SET GO
* FNCT1=RCV
* LED OFF
C
MDR11W-A/F4.4.5-2A

Figure 4.4.5-2A. DMA Receive (RCV), Program Controlled Transmit (XMT) Wraparound Test Mode Program, Functional Flow Diagram, Longword



MDR11W-A/F4.4.5-2B

Figure 4.4.5-2B. DMA Receive (RCV), Program Controlled Transmit (XMT) Wraparound Test Mode Program, Functional Flow Diagram, Longword (Continued)

BIT 2 - SET INPUT BYTE SWAP. The set input byte swap bit allows transposition of bytes within each word received from the other VMIVME-DR11W-A compatible interface as each word is loaded into the Input Data Register. If set to a logic "zero", no input byte swapping will occur, and input data will be loaded into the Input Data Register from the DI15 through DI00 bits of the interface ribbon cable.

If asserted to a logic "one", the set input byte swap bit will assert the BYTE SWAP MODE, causing the bytes received as bits DI15 through DI08 and DI07 through DI00 on the ribbon cable interface to be swapped prior to being written to the Input Data Register. For example, when receiving the first word during 32-bit Longword transfers, input data bits DI15 through DI08 will be loaded into Input Data Register bit positions 23 through 16 (or 07 through 00, if SET INPUT WORD SWAP is set), and data bits DI07 through DI00 will be loaded as Input Data Register bits 31 through 24 (or as bits 15 through 8, if SET INPUT WORD SWAP is asserted)

Figure 4.4.5-3 illustrates the input byte swap function as applied to the 16-bit I/O Data Register (IODR). Input byte swap formatting of the 32-bit IDR is illustrated in Figures 4.4.5-1 and 4.4.5-2.

BIT 1 - SET OUTPUT WORD SWAP. This bit allows control of the sequence in which the two words of the 32-bit longword Output Data Register are enabled to the interface cable signals DO15 through DO00. The output word swap will become operational if the VMIVME-DR11W-A interface is in a 32-bit TRANSMIT MODE with the /LONGWORD bit of the AMR asserted to a logic "zero", and the Other Side Ready for Transfer status bit asserted, and the FUNCTION 1 bit set to a logic "zero", or if 32-bit output transfers are specified in a test mode with the TEST MODE OUT bit asserted.

If set to a logic "zero", the first word enabled to the DR11W interface cable as an output word by the VMIVME-DR11W-A will be the High Word Output Data Register bits (ODR bits D31 through D16). The second word transmitted will be the Low Word Output Data Register bits (ODR bits D15 through D00). If byte swapping has not been also specified, the bit order within each 16-bit word will be maintained. Refer to Figure 4.4.5-4 for an illustration of the data bit organization with the SET OUTPUT WORD SWAP bit NOT set.

If set to a logic "one", the order in which the two words of the Output Data Register are enabled to the VMIVME-DR11W-A interface cable is reversed (or swapped). The first word enabled will be the Low Word Output Data Register bits, ODR bits D15 through D00. The High Word Output Data Register bits, ODR bits D31 through D16, will be the second word enabled, as directed by the load pointer. Figure 4.4.5-5 illustrates this word reversal. Again, unless byte swapping has been specified, the bit order within each 16-bit word is maintained. If byte swapping is specified, the byte position within each 16-bit output word will be swapped as described below.

BIT 0 - SET OUTPUT BYTE SWAP. The SET OUTPUT BYTE SWAP bit controls ordering of bytes within an output data word during output to the other VMIVME-DR11W-A compatible interface. If set to a logic "zero", no output byte swapping will occur for output data transfers.

If asserted to a logic "one", the SET OUTPUT BYTES SWAP bit will assert the BYTE SWAP MODE causing transposition of the bytes during output transfers. Thus data from the Output Register are byte swapped as they are enabled to DO15 through DO08 and DO07 through DO00. For example, when transmitting 32-bit longwords, data written as ODR bits 31 through 24 (or as bits 15 through 08, if output word swap is asserted) are enabled to bit positions DO07 through DO00 and bits written as ODR bits 23 through 16 (or as bits 07 through 00 if output word swap is asserted) are enabled as DO15 through DO08 in the first word output on the ribbon interface cable.

Figure 4.4.5-3 illustrates the output byte swap function for the 16-bit I/O Data Register. Output byte swapping from the 32-bit ODR is illustrated in Figures 4.4.5-4 and 4.4.5-5.

4.4.6 Board Status Register (BSR)

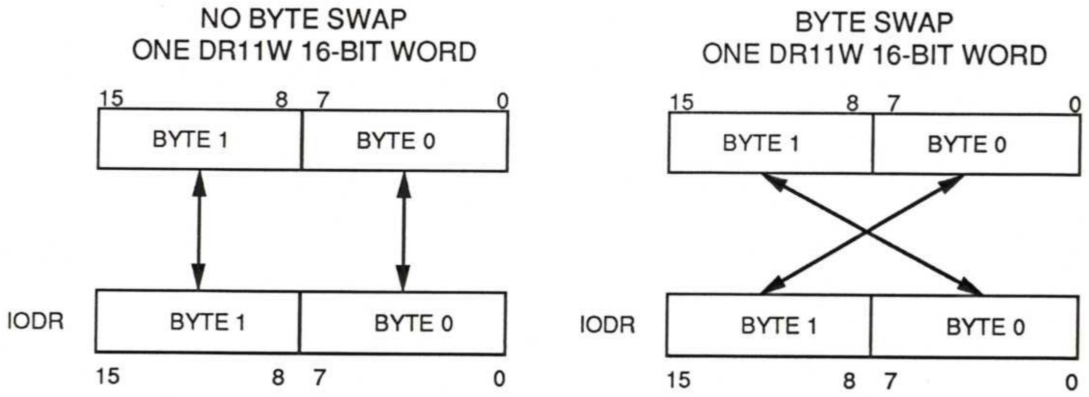
The BSR is an 8-bit read only register that allows the program to determine the status of various on-board and off-board I/O process events. The Board Status bits are registered, and are frozen in time during a read to prevent transitions on the VMEbus during the read window. All BSR status bits are cleared to a logic "zero" by either a Master Reset command issued via the BCR or by a VMEbus System Reset.

BITS 7 THROUGH 5 ARE NOT USED AND ARE SET TO "0".

BIT 4 - OTHER SIDE RDY FOR XFER (Other Side Ready for Transfer). This bit, if jumper JF is installed in the ENABLE position, indicates whether the other VMIVME-DR11W-A compatible interface has transitioned its READY H signal from High to Low to indicate that its GO bit has been set and that it is ready for the next group of data transfers to occur. In the DMA MODE with the GO bit set, I/O data transfers will be inhibited until this is asserted.

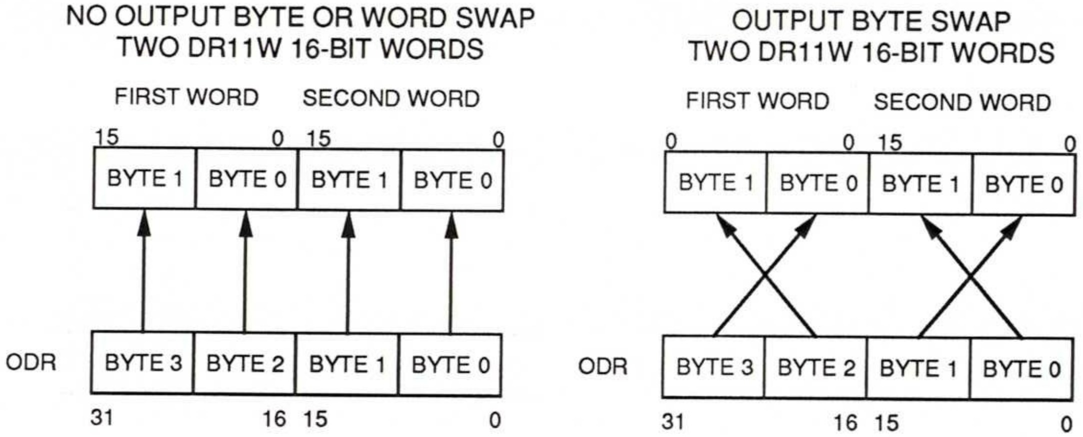
If jumper JF is installed in the DISABLE position, the OTHER SIDE RDY FOR XFER status bit will be forced to an asserted state. This, in effect, disables the use of the READY H signal during data transfers.

BIT 3 - SET OUT CYCLE REQ (Set Output Cycle Request). The SET OUT CYCLE REQ status bit will be set to a logic "zero" when an output transfer has occurred to the ODR (or the IODR) for 16-bit word operations and after the first word output has been acknowledged with a returned BUSY (Cycle Request A or B). This signifies that the ODR has been written with a new data word (or longword). This bit will be set to a logic "one" as a new output cycle is being requested and will return to a logic "zero" as the next write to the ODR completes. This bit can be used as an indicator of output data transfer activity.



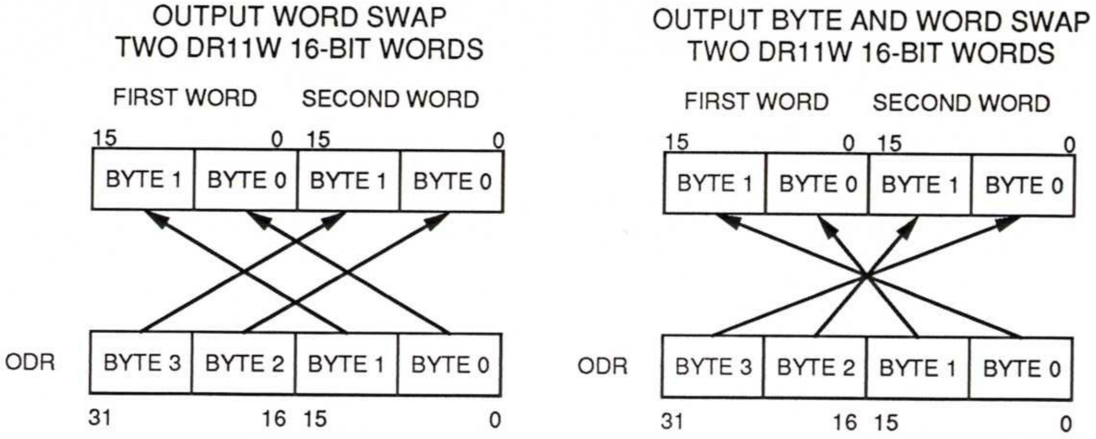
MDR11W-A/F4.4.5-3

Figure 4.4.5-3. 16-Bit I/O Data Register Formats



MDR11W-A/F4.4.5-4

Figure 4.4.5-4. Output Data Register *WITHOUT* Word Swap



MDR11W-A/F4.4.5-5

Figure 4.4.5-5. Output Data Register *WITH* Word Swap

The SET OUT CYCLE REQ status flag will be enabled, if the OTHER SIDE RDY FOR XFER bit is asserted, and the BCR FUNCTION 1 bit is set to a logic "zero" (selecting this board as a transmitter), or if the TEST MODE OUT bit is asserted. When enabled, it can be asserted by each occurrence, if 16-bit DATA MODE, or by the second occurrence, if 32-bit DATA MODE, of the Cycle Req A signal, the Cycle Req B signal or the Set Cycle Req command, depending on the user configuration.

BIT 2 - INPUT DATA READY. The INPUT DATA READY status bit, when asserted, is an indicator that new input data has been loaded into the Input Data Register, and that an input data transfer can be initiated to read the data. If DMA MODE is active with the GO and FUNCTION 1 bits asserted, a DMA request to write the data to memory will automatically be initiated (if enabled). In the programmed I/O MODE, the assertion of this bit indicates that data are ready for a programmed input transfer.

BIT 1 - BUS T.O. ERR (Bus Time-Out Error). The Bus Time-out Error flag will be set as the WDT times out (18.3 ms) if the VMIVME-DR11W-A interface, while operating in the DMA MODE, fails to receive a DTACK assertion within the WDT time-out period after the leading edge of the VMIVME-DR11W-A data strobe as described in (c.) of the description of the Bit 8 - ENABLE WDT control bit of the MCR in paragraph 4.4.5. See the NOTE after the description of the WDT T.O. ERR bit that follows.

The BUS T.O. ERR bit can be reset to a logic "zero" by writing a logic "zero" to the ENABLE WDT bit of the MCR after the Board Status read is completed. It is also reset when the ENA WDT bit of the Master Control Register is set to a logic "zero" during a Master Reset command or during the assertion of the VMEbus signal RESET* during power-up.

BIT 0 - WDT T.O. ERR (Watchdog Time-out Error). The Watchdog Time-out Error flag will be set (if enabled) if the Watchdog Timer times out (18.3 ms) because of either of the two conditions (a) or (b) detailed in the description of the Bit 8 - ENABLE WDT control bit of the MCR in paragraph 4.4.5. This bit, if set, may be reset to a logic "zero" by writing a logic "zero" to the ENABLE WDT bit of the MCR after the Board Status read command is completed. It is also cleared to a logic "zero" when the ENA WDT bit of Master Control Register is set to a logic "zero" by a Master Reset command and at power-up by the assertion of the VMEbus RESET* signal.

NOTE

THE VMIVME-DR11W-A HARDWARE DESIGN IS SUCH THAT IF THE WDT ERROR STATUS AND THE BUS TIME-OUT ERROR STATUS OCCUR CONCURRENTLY, AS WILL OFTEN BE THE CASE IN THE EVENT OF A WDT ERROR, THE FIRST ERROR TO BECOME ASSERTED WILL LOCKOUT (OR PREVENT) THE OCCURRENCE OF THE OTHER ERROR. THUS, ERROR IDENTIFICATION AMBIGUITY IS AVOIDED.

4.4.7 Set Cycle Req (SCRQ)

Under normal operating conditions in the DMA OUTPUT MODE, the next output cycle request is initiated by either the Cycle Req A or the Cycle Req B interface signal input, depending on the user setup and/or the position of jumper JB. The SET CYCLE REQ command ($006A), when enabled by the ENABLE PROGRM SET CYC REQ bit of the MCR (see bit 14 of paragraph 4.4.5), allows the program to set the SEND NEXT OUTPUT flip-flop, requesting the next output data word transfer cycle.

If LWORDL is not asserted, a DMA request for more output data will be generated (if not DONE) for each SET CYCLE REQ command issued.

If LWORDL is asserted, the first SET CYCLE REQ command will cause the second 16-bit word of the 32-bit Output Register to be output on the interface cable bits DO15 through DO00. A second SET CYCLE REQ command will generate a DMA request for more 32-bit output data. The SET OUT CYCLE REQ status bit of the BSR can be tested at the appropriate times to determine when each DMA request starts and completes.

A Cycle Request for Output Data will be set if the ODR (or the IODR) has been loaded with a new output data word, and :

a. TEST MODE OUT is set or
b. if TEST MODE OUT is NOT set and FUNCTION 1 is specifying the TRANSMIT MODE (a logic "zero"), and the Other Side Ready for Transfer bit has been asserted as a result of:
   - either the Enable Other BD Ready L jumper JF having been installed in the "Disable" position, or
   - with the Enable Other BD Ready L jumper JF installed in the "Enable" position, by the high to low transition of the Other Bd Ready signal from the other DR11W board on the A00 H interface signal input to the VMIVME-DR11W-A,

Figure 4.4.7-1 shows a Functional Flow Diagram of an example DMA transmit test program utilizing the SET CYCLE REQ command.

4.4.8 Load Input Data Register (LDIDR)

The Load Input Data Register (LDIDR) command allows the Input Data Register to be loaded under program control with a known data pattern for BIT verification in a test mode. For this mode to become operational, the ENABLE PROG LD IN DTA REG bit of the MCR must be asserted, along with the TEST MODE IN bit. The LDIDR function supports 16-bit word and 32-bit longword writes only. If a 32-bit capability does not exist in the user system, a 16-bit word LDIDR

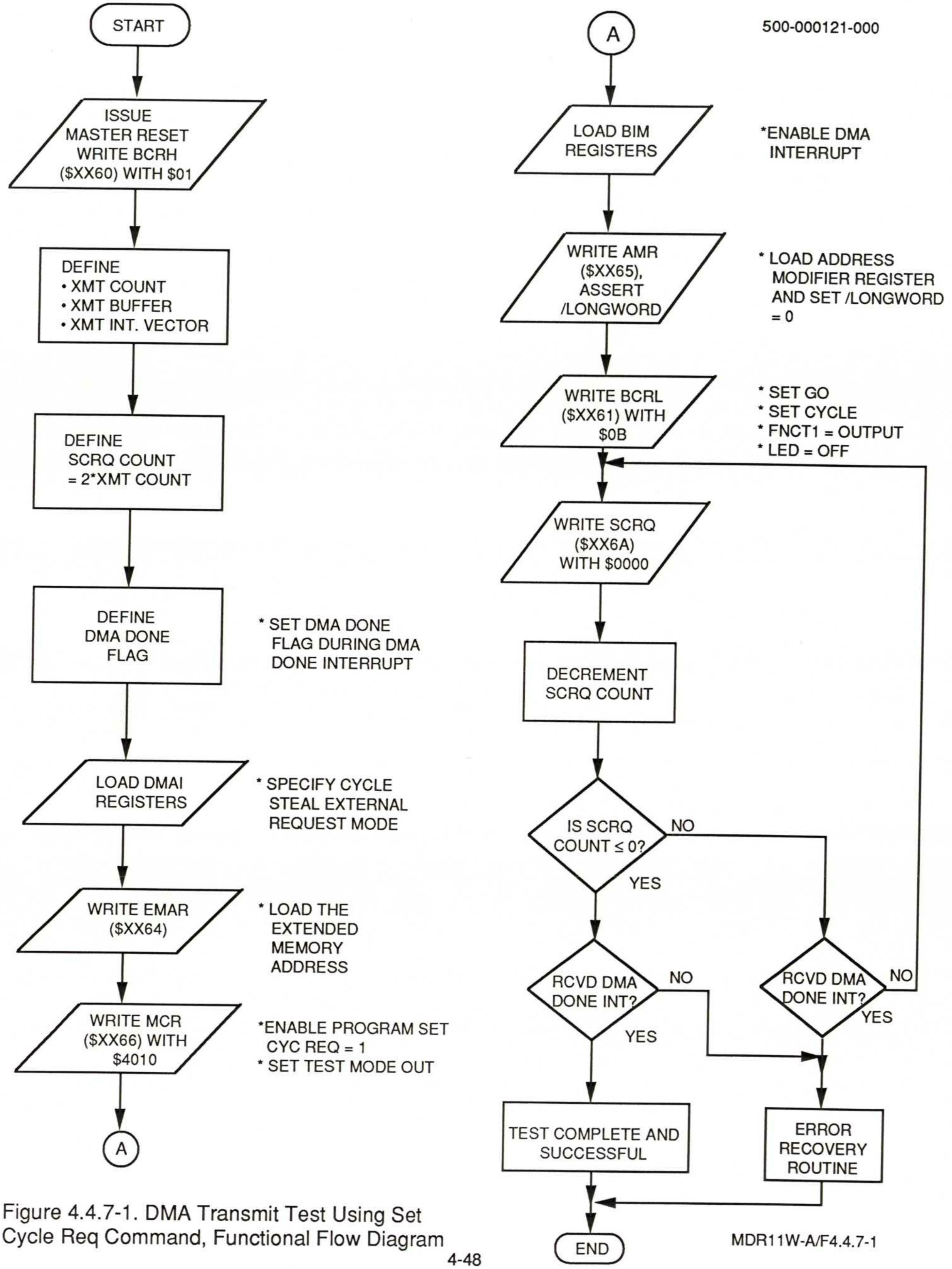

START
ISSUE MASTER RESET WRITE BCRH ($XX60) WITH $01
DEFINE
• XMT COUNT
• XMT BUFFER
• XMT INT. VECTOR
DEFINE SCRQ COUNT = 2*XMT COUNT
DEFINE DMA DONE FLAG
* SET DMA DONE FLAG DURING DMA DONE INTERRUPT
LOAD DMAI REGISTERS
* SPECIFY CYCLE STEAL EXTERNAL REQUEST MODE
WRITE EMAR ($XX64)
* LOAD THE EXTENDED MEMORY ADDRESS
WRITE MCR ($XX66) WITH $4010
*ENABLE PROGRAM SET CYC REQ = 1
* SET TEST MODE OUT
A
A
LOAD BIM REGISTERS
*ENABLE DMA INTERRUPT
WRITE AMR ($XX65), ASSERT /LONGWORD
* LOAD ADDRESS MODIFIER REGISTER AND SET /LONGWORD = 0
WRITE BCRL ($XX61) WITH $0B
* SET GO
* SET CYCLE
* FNCT1 = OUTPUT
* LED = OFF
WRITE SCRQ ($XX6A) WITH $0000
DECREMENT SCRQ COUNT
IS SCRQ COUNT ≤ 0?
NO
YES
RCVD DMA DONE INT?
NO
YES
RCVD DMA DONE INT?
NO
YES
TEST COMPLETE AND SUCCESSFUL
ERROR RECOVERY ROUTINE
END
MDR11W-A/F4.4.7-1

Figure 4.4.7-1. DMA Transmit Test Using Set Cycle Req Command, Functional Flow Diagram

can be followed by a 16-bit read of the IODR at address ($XX62) and can be used to verify the register as a single 16-bit word Input Data Register.

In the 32-bit longword mode, the Input Data Register must **always** be loaded with two successive 32-bit longword writes to load both words of the 32-bit register, and to reinitialize the load pointer. This is because the 32-bit IDR is loaded from the 32-bit LDIDR command, 16-bits at a time. As the LDIDR command loads the IDR, the Input Data Ready status bit will be set in the BSR to indicate that data is available in the IDR. It will be asserted during the second LDIDR of a 32-bit longword IDR load operation, and once for every LDIDR of a 16-bit word IDR load operation.

If the SET INPUT WORD SWAP bit is not asserted in the MCR, the first LDIDR command (write) will load the associated upper 16-bit data word into the High Word Input Data Register. A second identical LDIDR command will then load the lower 16-bit data word to the Low Word Input Data Register. A 32-bit read of the Input Data Register at address ($XX78) can then be used to verify the contents of the 32-bit Input Data Register while still in the TEST MODE. Figure 4.4.8-1 illustrates this two command load process.

Asserting the SET INPUT WORD SWAP bit in the MCR will reverse the pointer action in that the first word written with a longword LDIDR command will load the Low Word Input Data Register, and the second word written will load the High Word Input Data Register. Refer to Figure 4.4.8-2 for an illustration of the load process with the SET INPUT WORD SWAP bit asserted. Functional Flow Diagrams of sample test mode programs have been developed as examples of how this BIT function might be utilized. These examples test the integrity of the IDR and the function of the Input Data Ready status and are included as Figure 4.4.8-1 (A and B) for a 16-bit LDIDR command, and Figure 4.4.8-2 (A and B) for the 32-bit LDIDR operation.

4.4.9 VMIVME-DR11W-A 32-Bit Output Data Register (ODR)

The 32-bit ODR is written by the CPU either when operating as bus master (DMA transfer) or under program control. The outputs of the ODR are enabled 16-bits at a time via high-speed bus drivers toward the external VMIVME-DR11W-A (or VMIVME-DR11W-A compatible) interface during an output transfer. The ODR must be written as a 32-bit longword only.

BITS 31 THROUGH 0 - OUTPUT DATA D31 THROUGH D0 (Output Data bits D31 through D0).

4.4.10 VMIVME-DR11W-A 32-Bit Input Data Register (IDR)

The 32-bit IDR may be read under DMA control or under program control. It is capable of longword input transfers (i.e., 32-bits per transfer cycle). The Input Data Register receives and stores the data that are transferred in a two-word

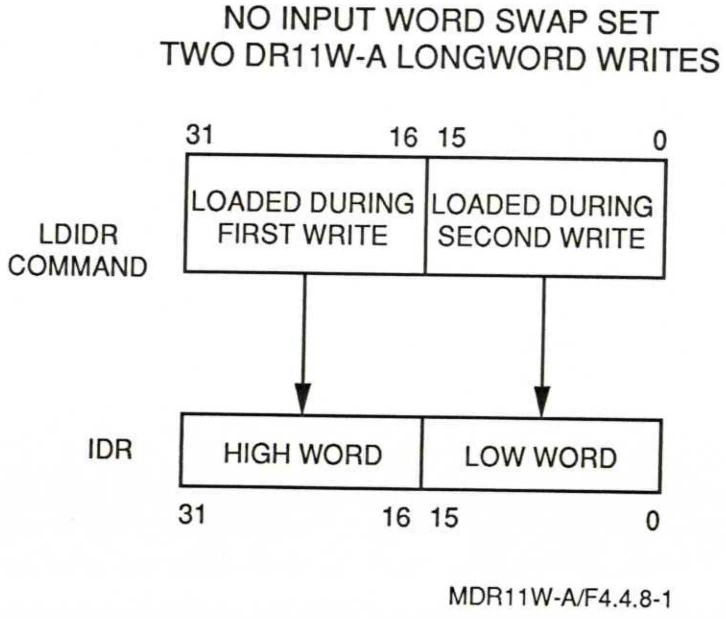


Figure 4.4.8-1. LDIDR Command Format *WITHOUT* Word Swap, 32-Bit

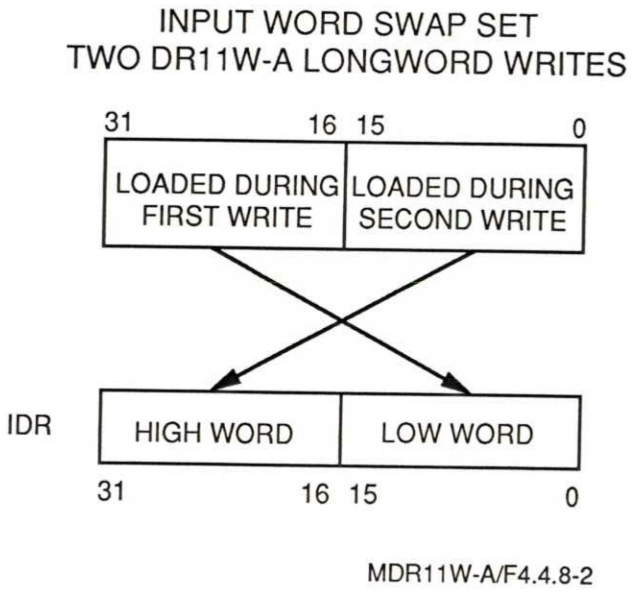

INPUT WORD SWAP SET
TWO DR11W-A LONGWORD WRITES
31
16
15
0
LOADED DURING FIRST WRITE
LOADED DURING SECOND WRITE
IDR
HIGH WORD
LOW WORD
31
16
15
0
MDR11W-A/F4.4.8-2

Figure 4.4.8-2. LDIDR Command Format *WITH* Word Swap, 32-Bit

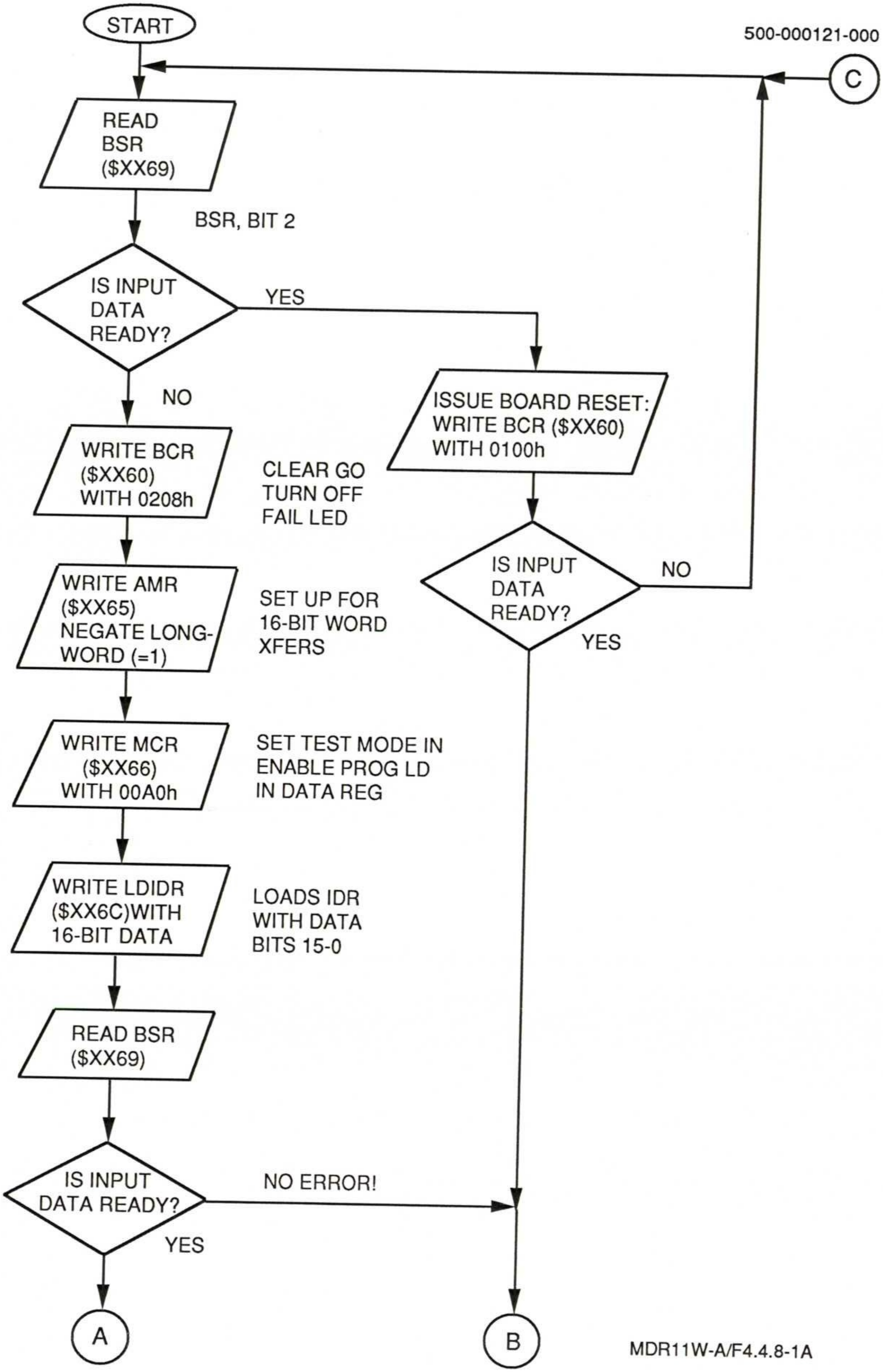


Figure 4.4.8-1A. 16-Bit LDIDR Bit Program Functional Flow Diagram

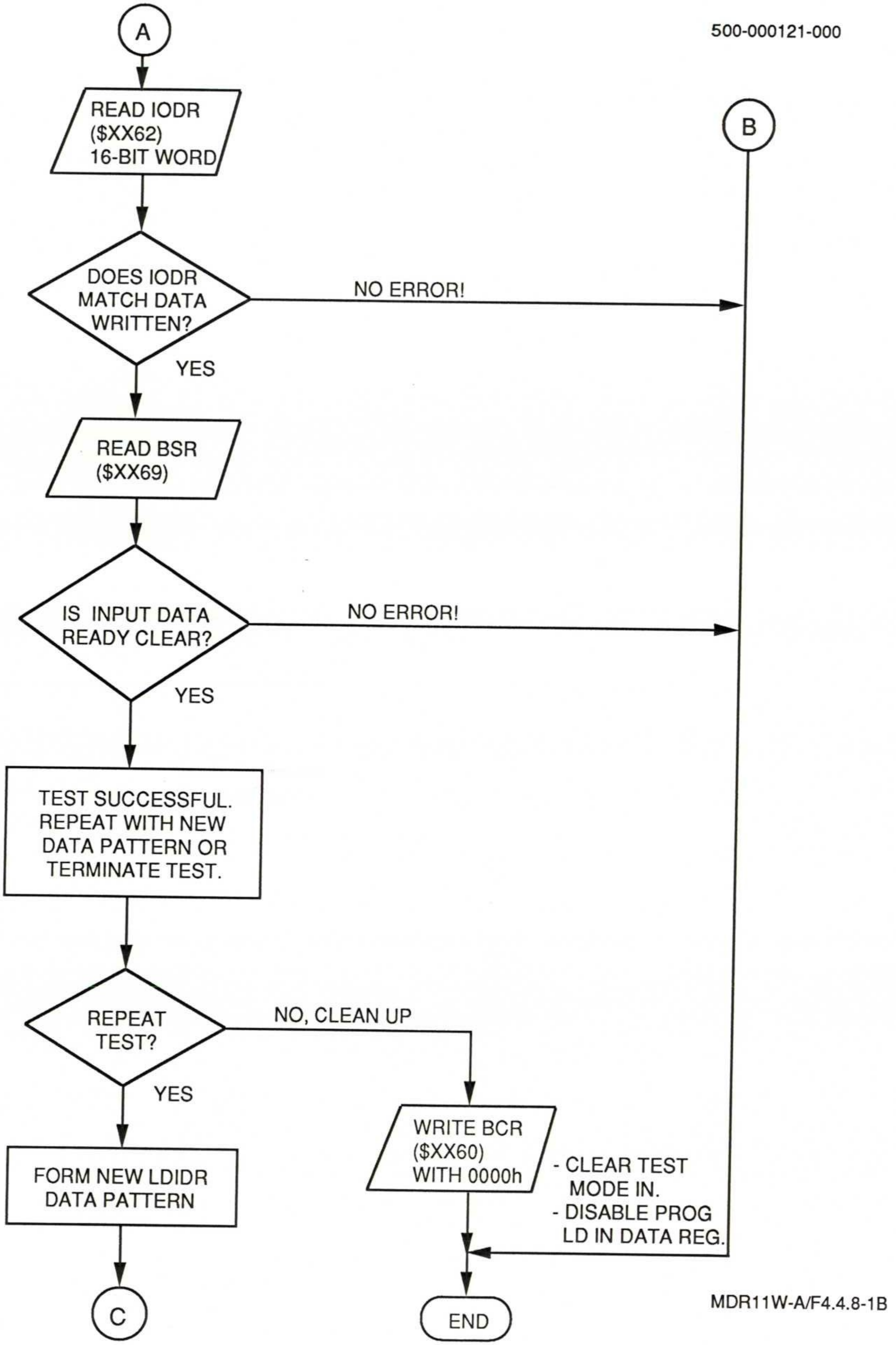


Figure 4.4.8-1B. 16-Bit LDIDR Bit Program Functional Flow Diagram (Continued)

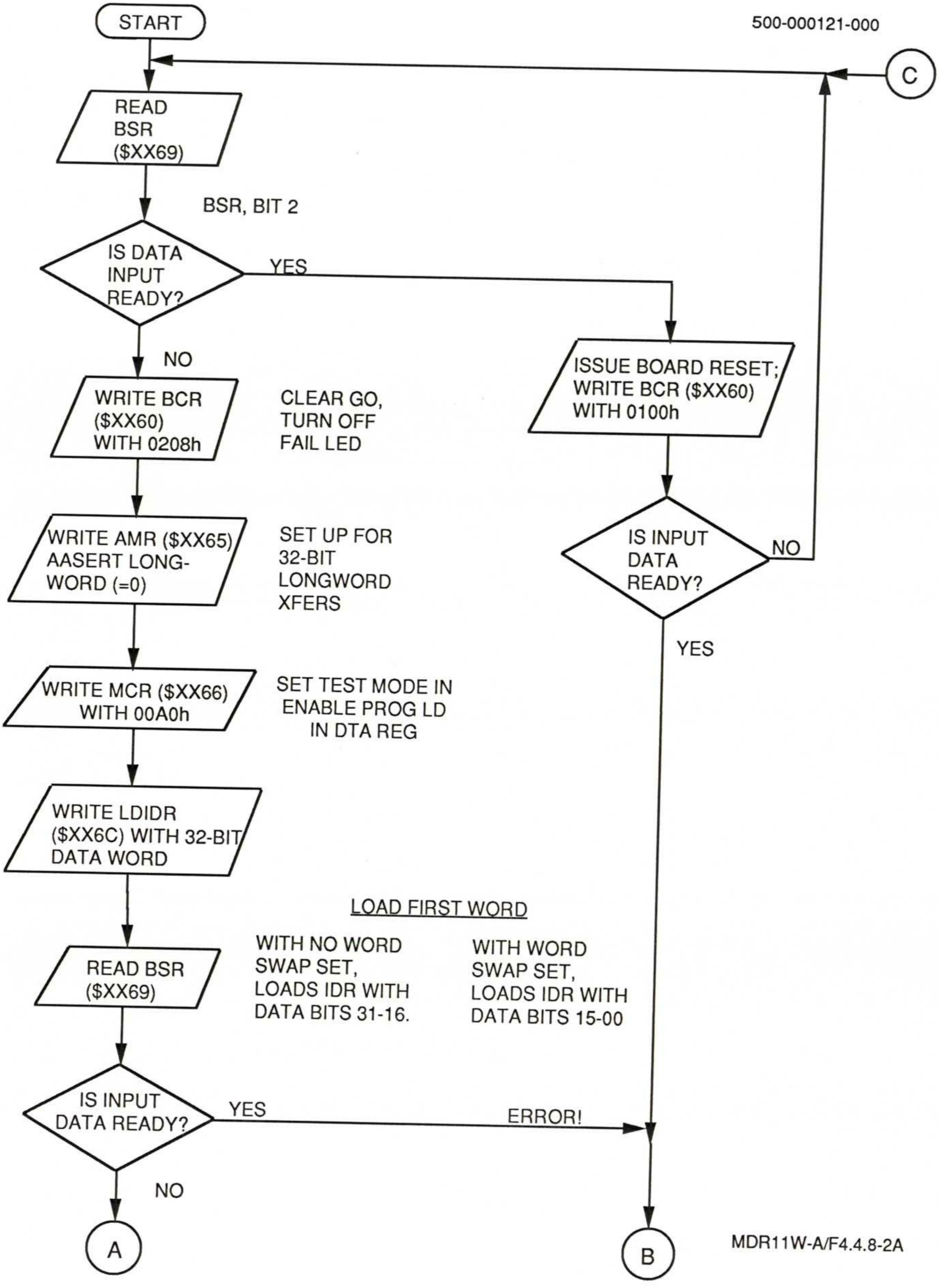
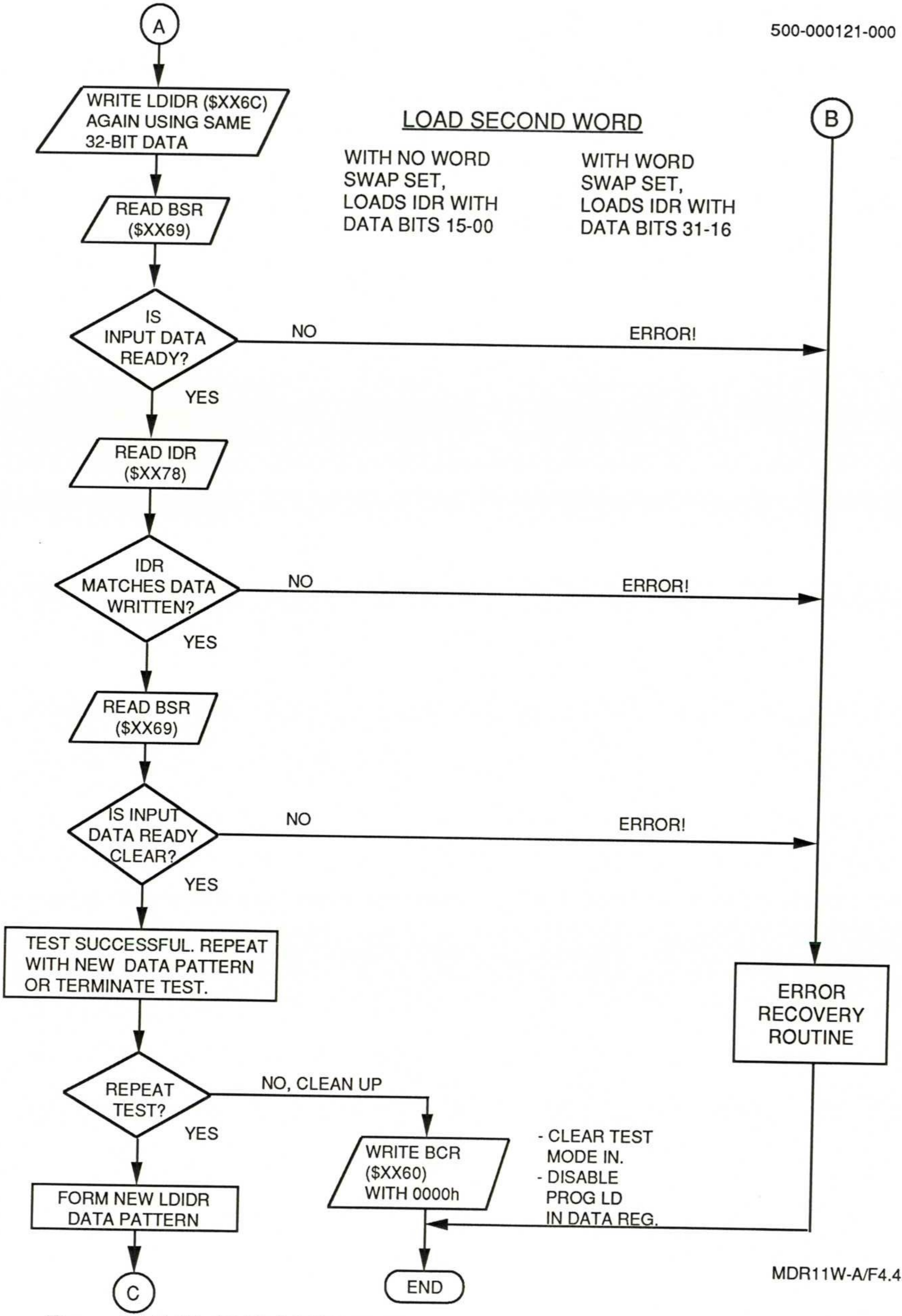

START
READ BSR ($XX69)
BSR, BIT 2
IS DATA INPUT READY?
YES
NO
WRITE BCR ($XX60) WITH 0208h
CLEAR GO, TURN OFF FAIL LED
WRITE AMR ($XX65) AASERT LONG-WORD (=0)
SET UP FOR 32-BIT LONGWORD XFERS
WRITE MCR ($XX66) WITH 00A0h
SET TEST MODE IN ENABLE PROG LD IN DTA REG
WRITE LDIDR ($XX6C) WITH 32-BIT DATA WORD
LOAD FIRST WORD
READ BSR ($XX69)
WITH NO WORD SWAP SET, LOADS IDR WITH DATA BITS 31-16.
WITH WORD SWAP SET, LOADS IDR WITH DATA BITS 15-00
IS INPUT DATA READY?
YES
ERROR!
NO
A
ISSUE BOARD RESET; WRITE BCR ($XX60) WITH 0100h
IS INPUT DATA READY?
NO
YES
B
C
MDR11W-A/F4.4.8-2A

Figure 4.4.8-2A. 32-Bit LDIDR Bit Program Functional Flow Diagram


A
WRITE LDIDR ($XX6C) AGAIN USING SAME 32-BIT DATA
LOAD SECOND WORD
WITH NO WORD SWAP SET, LOADS IDR WITH DATA BITS 15-00
WITH WORD SWAP SET, LOADS IDR WITH DATA BITS 31-16
B
READ BSR ($XX69)
IS INPUT DATA READY?
NO
ERROR!
YES
READ IDR ($XX78)
IDR MATCHES DATA WRITTEN?
NO
ERROR!
YES
READ BSR ($XX69)
IS INPUT DATA READY CLEAR?
NO
ERROR!
YES
TEST SUCCESSFUL. REPEAT WITH NEW DATA PATTERN OR TERMINATE TEST.
ERROR RECOVERY ROUTINE
REPEAT TEST?
NO, CLEAN UP
YES
WRITE BCR ($XX60) WITH 0000h
- CLEAR TEST MODE IN.
- DISABLE PROG LD IN DATA REG.
FORM NEW LDIDR DATA PATTERN
C
END

MDR11W-A/F4.4.8-2B

Figure 4.4.8-2B. 32-Bit LDIDR Bit Program Functional Flow Diagram (Continued) sequence from the external VMIVME-DR11W-A (or compatible) interface during an input transfer. The IDR is loaded with 16-bits at a time, as clocked, depending on the user configuration, by the Cycle Req A (or the Cycle Req B) signal from the other VMIVME-DR11W-A compatible interface, or by the Set Cycle Req command.

The IDR can also be written under program control in a test mode. Refer to the description of the Load Input Data Register command in paragraph 4.4.7 for a description of that procedure.

4.5 SEQUENCE OF REGISTER LOADING

The receiving VMIVME-DR11W-A compatible interface must be set up before the transmitting interface to avoid losing the first word transferred, if jumper JF is disabled.

To transfer a block of data the programming sequence of the VMIVME-DR11W-A should be as follows:

a. Load the registers in the MC68153 and SCB68430.
b. The CCR should be the last register loaded in the SCB68430.
c. Load the AMR and EMAR.
d. Load the BCR with the GO bit (bit 0) and the CYCLE bit (bit 1) (if transmitting) set to "zero" (this step needed only for assurance)
e. Reload the BCR with the GO and CYCLE bits set to "one".

Note that the start bit (CCR bit 07) must be set inside in SCB68430 before the GO and CYCLE bits are set in the BCR. The start bit in the SCB68430 is independent of the CYCLE (start) bit in the BCR. The C1 interface signal input (BCR - bit 10) controls the direction of the I/O Data Register, while the direction bit in the SCB68430 (OCR - bit 07) controls the transfer direction of the SCB68430. These two bits must be in the same logic state for correct operation.

An example of register initialization sequence for transmitting data (from memory) is shown in Table 4.5-1. After the first DMA transfer is complete, the VMIVME-DR11W-A can be reinitialized with only seven steps to transmit a second block of data (see Table 4.5-2).

An example of a register initialization sequence for receiving data (transfer to memory) is shown in Table 4.5-3. After the first block transfer is complete, the VMIVME-DR11W-A can be reinitialized with only eight steps to receive a second block of data (see Table 4.5-4).

4.6 SAMPLE SOFTWARE LISTINGS

VMIC has provided sample software listings in Appendix C to assist the user in programming the VMIVME-DR11W-A. Refer to Appendix C for an itemized list of software listings provided.

Table 4.5-1. Register Initialization Sequence for Transmitting a 4 k Word Block Starting Data Address $40000

STEP	REGISTER LOCATION	REGISTER NAME	REGISTER ADDRESS OFFSET	DATA LOADED (HEXADECIMAL)	COMMENT
1	68430	DCR	04	B8	Cycle Steal
2	68430	OCR	05	12	Word from memory
3	68430	MTCH	0A	1000	16-bit transfer count
4	68430	MACMH	0D	04	Upper 8 bits of memory address
5	68430	MACML	0E	0000	Lower 16 bits of memory address
6	68430	DIVR	25	42	Done interrupt vector
7	68153	AIVR	4D	40	Attention interrupt vector
8	68153	AICR	45	1F	Enable internal vector with auto clear
9	68153	DICR	47	3F	Enable external vector with auto clear
10	On-Board	EMAR AND AMR	64	0079	Word, ADDR MOD=39
11	68430	CSR	00	B9	Clear channel Status Register
12	68430	CCR	07	88	Start DMA chip
13	On-Board	BCR	61	8F	Go, enable ATTN, LED OFF, Cycle, Transmit

MDR11W-A/T4.5-1

Table 4.5-2. Register Initialization Sequence for Transmitting a Second 4 K Word Block Starting Data Address $20000

STEP	REGISTER LOCATION	REGISTER NAME	REGISTER ADDRESS OFFSET	DATA LOADED (HEXADECIMAL)	COMMENT
1	68430	MTCH	0A	1000	16-bit transfer count
2	68430	MACMH	0D	02	Upper 8 bits of memory address
3	68430	MACML	0E	0000	Lower 16 bits of memory address
4	68153	DICR	47	3F	Enable external vector with auto clear
5	68430	CSR	00	B9	Register
6	68430	CCR	07	88	Start DMA chip
7	On-Board	BCR	61	8F	Go, enable ATTN, LED OFF, Cycle, Transmit

MDR11W-A/T4.5-2

Table 4.5-3. Register Initialization Sequence for Receiving a 4 k Word Block Starting Data Address $40000

STEP	REGISTER LOCATION	REGISTER NAME	REGISTER ADDRESS OFFSET	DATA LOADED (HEXADECIMAL)	COMMENT
1	68430	DCR	04	B8	Cycle Steal
2	68430	OCR	05	92	Word to memory
3	68430	MTCH	0A	1000	16-bit transfer count
4	68430	MACMH	0D	04	Upper 8 bits of memory address
5	68430	MACML	0E	0000	Lower 16 bits of memory address
6	68430	DIVR	25	42	Done interrupt vector
7	68153	AIVR	4D	40	Attention interrupt vector
8	68153	AICR	45	1F	Enable internal vector with auto clear
9	68153	DICR	47	3F	Enable external vector with auto clear
10	On-Board	EMAR AND AMR	64	0079	Word, ADDR MOD=39
11	68430	CSR	00	B9	Clear channel Status Register
12	68430	CCR	07	88	Start DMA chip
13	On-Board	BCR	61	AC	Go=0, Receive direction, enable ATTN, LED OFF
14	On-Board	BCR	61	AD	Go, Receive direction, enable ATTN, LED OFF

MDR11W-A/T4.5-3

Table 4.5-4. Register Initialization Sequence for Receiving a Second 4 K Word Block Starting Data Address $20000

STEP	REGISTER LOCATION	REGISTER NAME	REGISTER ADDRESS OFFSET	DATA LOADED (HEXADECIMAL)	COMMENT
1	68430	MTCH	0A	1000	16-bit transfer count
2	68430	MACMH	0D	02	Upper 8 bits of memory address
3	68430	MACML	0E	0000	Lower 16 bits of memory address
4	68153	DICR	47	3F	Enable external vector with auto clear
5	68153	AICR	45	1F	Enable internal vector with auto clear
6	68430	CSR	00	B9	Clear channel Status Register
7	68430	CCR	07	88	Start DMA chip
8	On-Board	BCR	61	AD	Go, receive direction, enable ATTN, LED OFF

MDR11W-A/T4.5-4

SECTION 5

CONFIGURATION AND INSTALLATION

5.1 UNPACKING PROCEDURES

CAUTION

SOME OF THE COMPONENTS ASSEMBLED ON VMIC's PRODUCTS MAY BE SENSITIVE TO ELECTROSTATIC DISCHARGE AND DAMAGE MAY OCCUR ON BOARDS THAT ARE SUBJECTED TO A HIGH ENERGY ELECTROSTATIC FIELD. WHEN THE BOARD IS TO BE LAID ON A BENCH FOR CONFIGURING, ETC., IT IS SUGGESTED THAT CONDUCTIVE MATERIAL BE INSERTED UNDER THE BOARD TO PROVIDE A CONDUCTIVE SHUNT. UNUSED BOARDS SHOULD BE STORED IN THE SAME PROTECTIVE BOXES IN WHICH THEY WERE SHIPPED.

Upon receipt, any precautions found in the shipping container should be observed. All items should be carefully unpacked and thoroughly inspected for damage that might have occurred during shipment. The board(s) should be checked for broken components, damaged printed circuit board(s), heat damage, and other visible contamination. All claims arising from shipping damage should be filed with the carrier and a complete report sent to VMIC together with a request for advice concerning disposition of the damaged item(s).

5.2 PHYSICAL INSTALLATION

CAUTION

DO NOT INSTALL OR REMOVE BOARD WHILE POWER IS APPLIED.

De-energize the equipment and insert the board into an appropriate slot of the chassis. While ensuring that the board is properly aligned and oriented in the supporting card guides, slide the board smoothly forward against the mating connector until firmly seated.

5.3 JUMPER INSTALLATIONS

The VMIVME-DR11W-A interface boards provide twelve groups of jumpers for jumper-selection of the various user-determined control options. The BUSY handshake signal polarity, the VMEbus priority level (bus request and bus grant levels), either standard or short I/O on-board register base addressing, the standard I/O address (if selected), the short I/O base address (if selected), Cycle Req A/Cycle Req B handshake optioning, Cycle Req A (or B) handshake response polarity, BUSY DELAY clock and the BUSY signal deskew delay are all jumper-selectable. Use of the LWORD* signal while operating as a VMEbus Master, recognition of the other board READY signal, and enabling or disabling of the Cycle Req A (or B) input signal delay are also jumper-determined. Refer to Figure 5.3-1 for jumper locations.

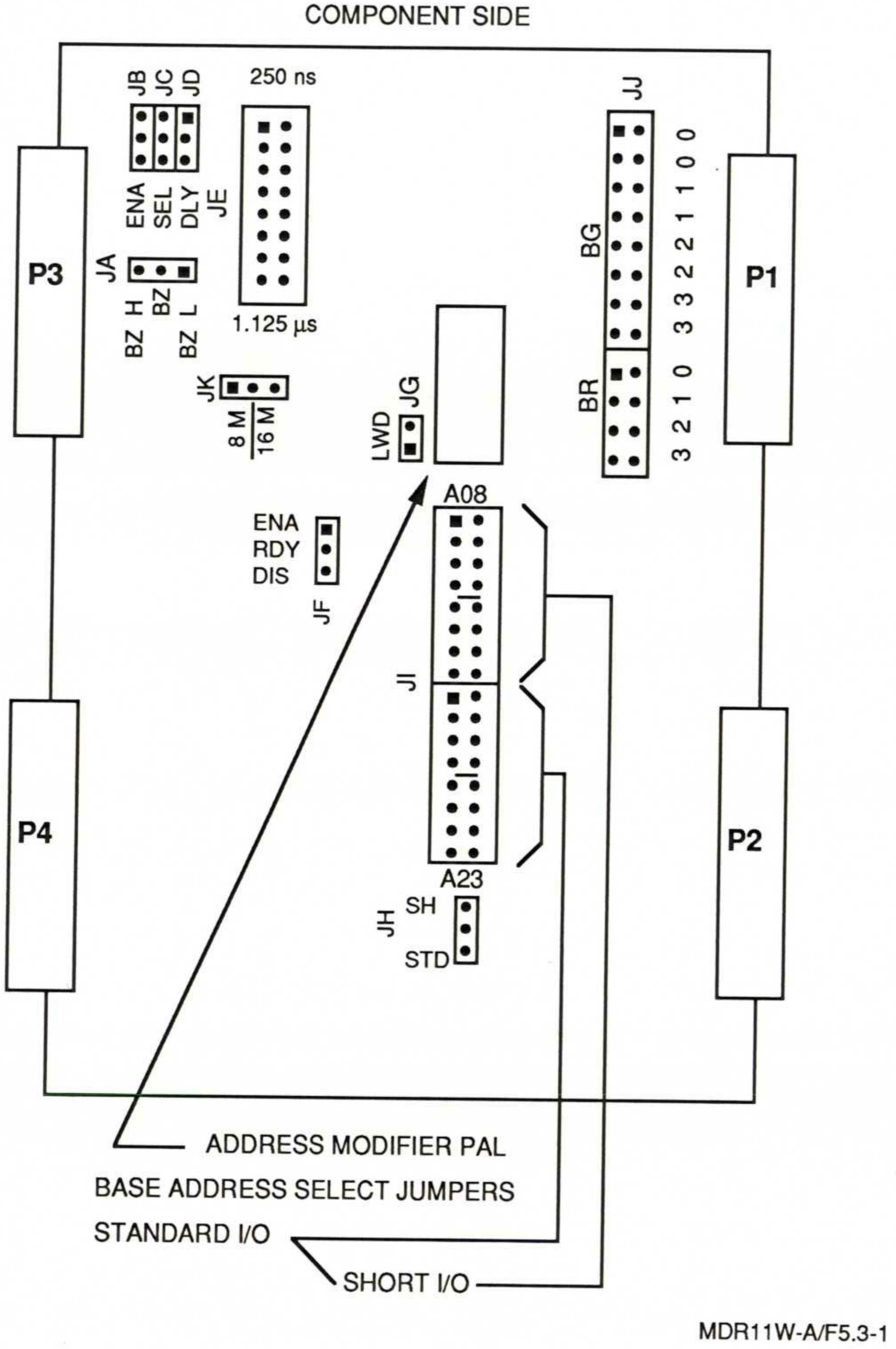
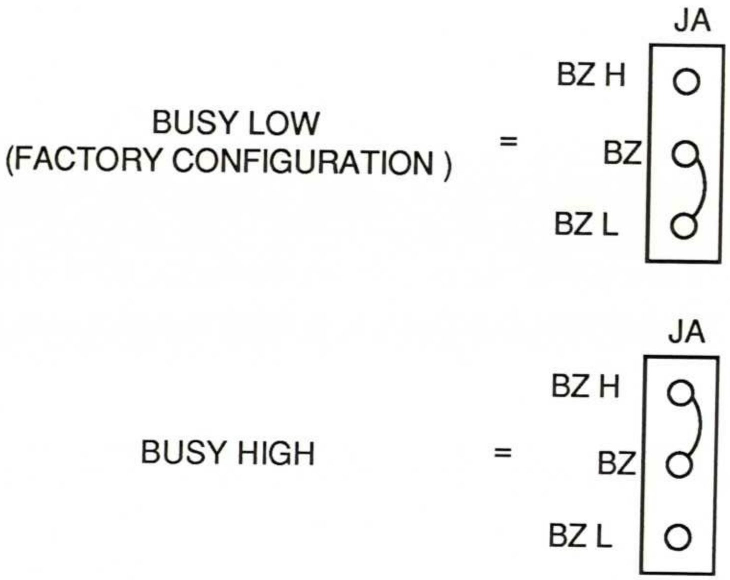


Figure 5.3-1. VMIVME-DR11W-A Configuration Information

5.3.1 Busy Polarity

The polarity of the BUSY (BZ) handshake signal can be selected to be active H or L using jumper JA. As shown in Figure 5.3.1-1, BZ L must be selected for back-to-back operation of two VMIVME-DR11W-A Boards. The factory configuration is BZ L.



MDR11W-A/F5.3.1-1

Figure 5.3.1-1. Busy Signal Polarity Selection, Busy L Must Be Selected for Back-to-Back Operation of Two VMIVME-DR11W Boards

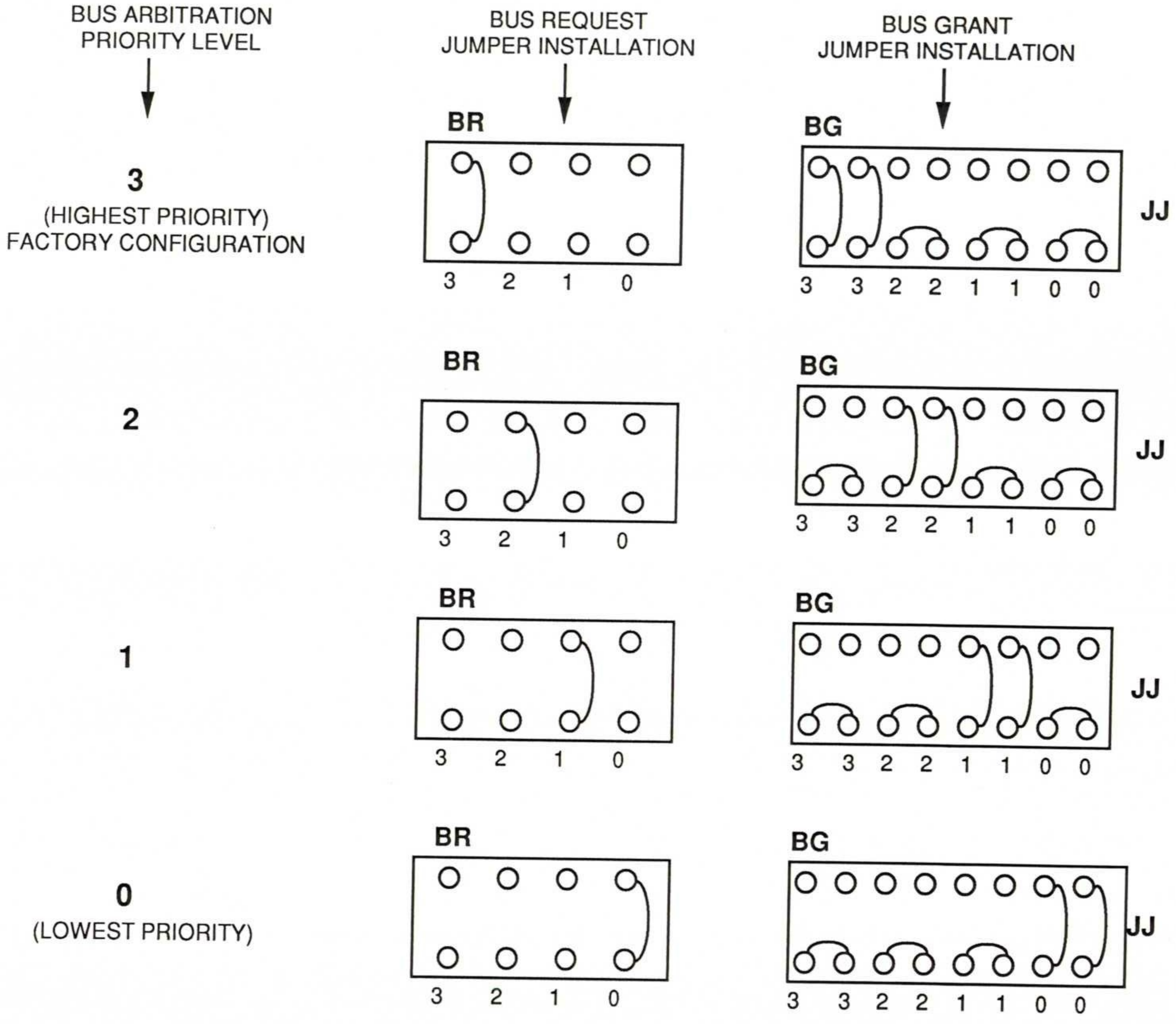
5.3.2 VMEbus Priority Jumpers

Jumper JJ provides for the selection of bus request level and grant level. The reader should refer to Figure 5.3.2-1 for selection of bus priority level (3, 2, 1, or 0). The factory configuration is level 3.

5.3.3 Board Base Address

The VMIVME-DR11W-A occupies 256 bytes of the VMEbus I/O address space. Using jumper JH the user can configure the board to locate the board base address in either the VMEbus short (SH) I/O space or the VMEbus standard (STD) I/O space. The factory configuration is SH.

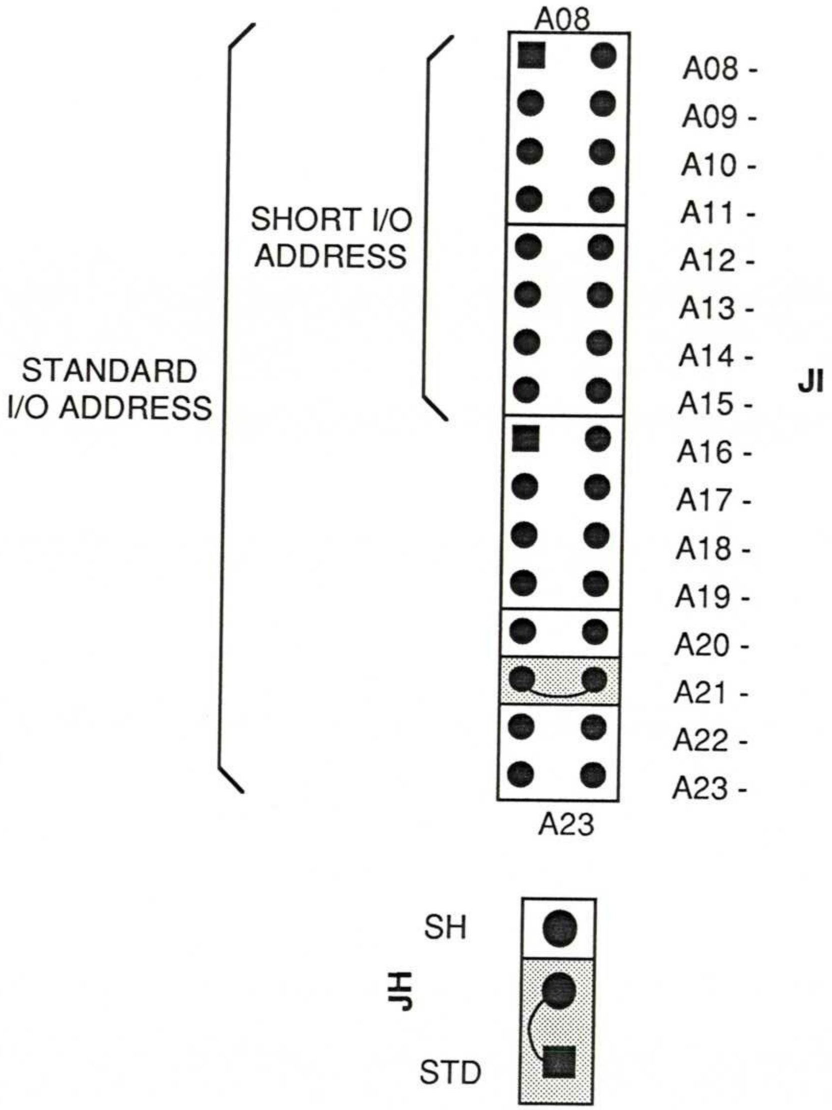
The upper sixteen bits of the STD I/O address are jumper-selectable. Jumper JI must be configured to select a STD I/O address space if


BUS ARBITRATION PRIORITY LEVEL
BUS REQUEST JUMPER INSTALLATION
BUS GRANT JUMPER INSTALLATION
3
(HIGHEST PRIORITY)
FACTORY CONFIGURATION
2
1
0
(LOWEST PRIORITY)
BR
3 2 1 0
BG
3 3 2 2 1 1 0 0
JJ

NOTE: THIS BOARD IS FACTORY CONFIGURED FOR LEVEL 3.

MDR11W-A/F5.3.2-1

Figure 5.3.2-1. Jumper Installation for Selection of VMEbus Priority Level



MDR11W-A/F5.3.3-1

Figure 5.3.3-1. Board Base Address Jumpers

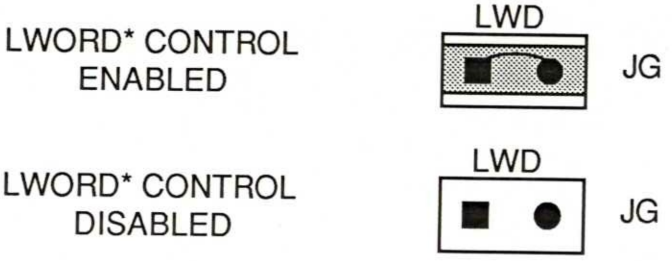
jumper JH has been configured for STD. Figure 5.3.3-1 shows a STD I/O address selection of $DFFFXX. The factory configuration is for address $XXFFE0XX.

If the SH I/O space is chosen with jumper JH, the upper 8 address bits (A16 through A23) selectable by jumper JI are ignored, and the short I/O address will be determined by the 8 jumpers of JI (A08 through A15, refer to Figure 5.3.3-1).

5.3.4 LWORD* Control Jumper

The LWD jumper has been provided to maintain compatibility with VMIC's earlier VMIVME-DR11W Board and should be removed for those 16-bit transfer only applications in which it is desirable for bit 6 of the Address Modifier Register to be set to a logic level other than that required for a 32-bit longword transfer (see paragraph 4.4.4). The LWD jumper, JG, when removed, causes the LONGWORD bit (bit 6) of the Address Modifier Register to be ignored. The LWORD* signal will always be output from the board unasserted (high) when LWORD* is enabled during DMA transfers to specify a 16-bit word transfer size.

When operating in a 32-bit DMA transfer application, this jumper must be installed so that the LONGWORD bit will be used to determine the state of the LWORD* address signal during DMA transfers. The upper half of Figure 5.3.4-1 shows this jumper installed. This is the factory selected configuration.



MDR11W-A/F5.3.4-1

Figure 5.3.4-1. LWD Jumper

5.3.5 Enable Other Board Ready Jumper

This jumper, JF, is a two position jumper that allows the user to either enable (ENA) or disable (DIS) the "Other Board Ready" feature of the VMIVME-DR11W-A Board. Figure 5.3.5-1 shows this jumper. The factory selected configuration is the ENA position.

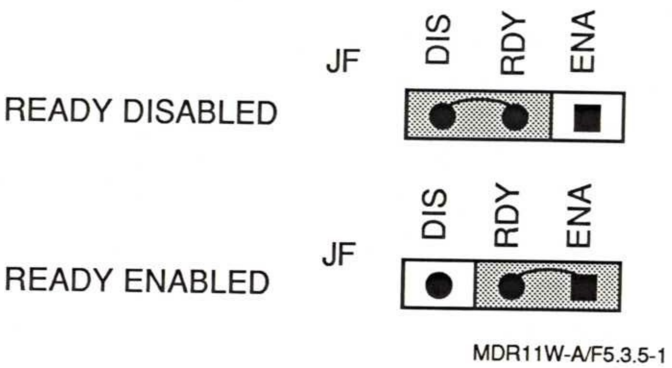


Figure 5.3.5-1. Other Board READY Jumper

The "Other Board Ready" feature of the VMIVME-DR11W-A Board, when enabled, establishes the condition that the transfer request circuitry will wait for the other board's READY H interface signal to transition from a High state to a Low state before allowing the CYCLE REQ A or the CYCLE REQ B input to be recognized as a valid transfer cycle request.

When disabled, the "Other Board Ready" feature causes the transfer request circuitry to ignore the other board's READY H signal when responding to a CYCLE REQ A (or B) transfer cycle request.

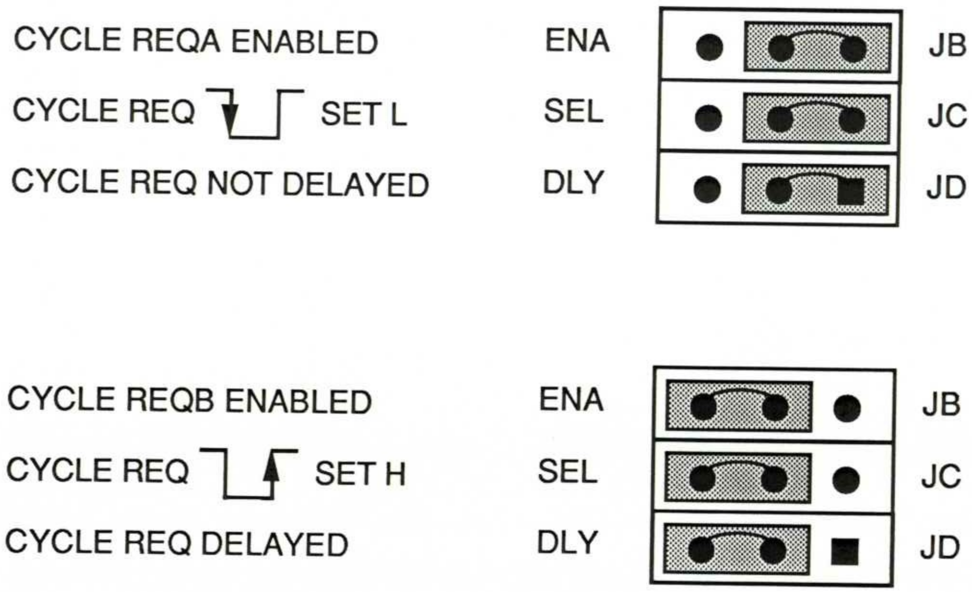
5.3.6 Cycle Request Jumpers

There are three jumpers that affect the operating modes of the Cycle Request logic, jumpers JB, JC, and JD. Refer to Figure 5.3.6-1 for these jumpers.

The enable (ENA) REQA or REQB jumper, JB, allows the user to enable either the CYCLE REQA H or the CYCLE REQB H input signal to be used as the active Cycle Request input signal. Thus, there is no requirement for the user to specially ground the inactive Cycle Request input. The factory setting is REQA enabled.

The select (SEL) H or L jumper, JC, selects between use of the leading (Low going) or the trailing (High going) edge of the enabled CYCLE REQ input signal to set the cycle request flip-flop of the VMIVME-DR11W-A interface. The factory setting is CYCLE REQUEST SET H.

The delay (DLY) jumper, JD, specifies that the active cycle request signal, CYCLE REQA (or CYCLE REQB), be delayed by at least one VMEbus SYS CLK clock cycle before being input to the cycle request logic of the VMIVME-DR11W-A interface. This feature may be invoked to improve noise immunity of the cycle request input circuitry, or to provide an extra clock cycle of skew delay for data presented from the other board's output data signals (DO15 through DO00) while this board is receiving data. The factory setting is CYCLE REQUEST not delayed.



MDR11W-A/F5.3.6-1

Figure 5.3.6-1. Cycle Request Jumper

5.3.7 Busy Delay Clock Jumper

The Busy Delay Clock jumper, JK, is a two position jumper that allows the user to select between two fundamental clock rates for determining the BUSY DELAY selection range. With the jumper in the 8M position, the input clock to the Busy Delay Counter is 8 MHz with a 125 ns (nanosecond) nominal period. This allows selection of a BUSY (deskew) DELAY ranging from 250 ns to 1125 ns (see Section 5.3.8). With the jumper on the 16M position, the Busy Delay Clock is selected to be 16 MHz with a 62.5 ns nominal period resulting in a selection range of 125 ns to 625 ns. Figure 5.3.7-1 shows this jumper with the 16M factory setting selected.

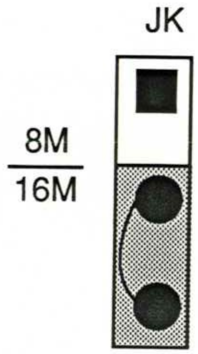


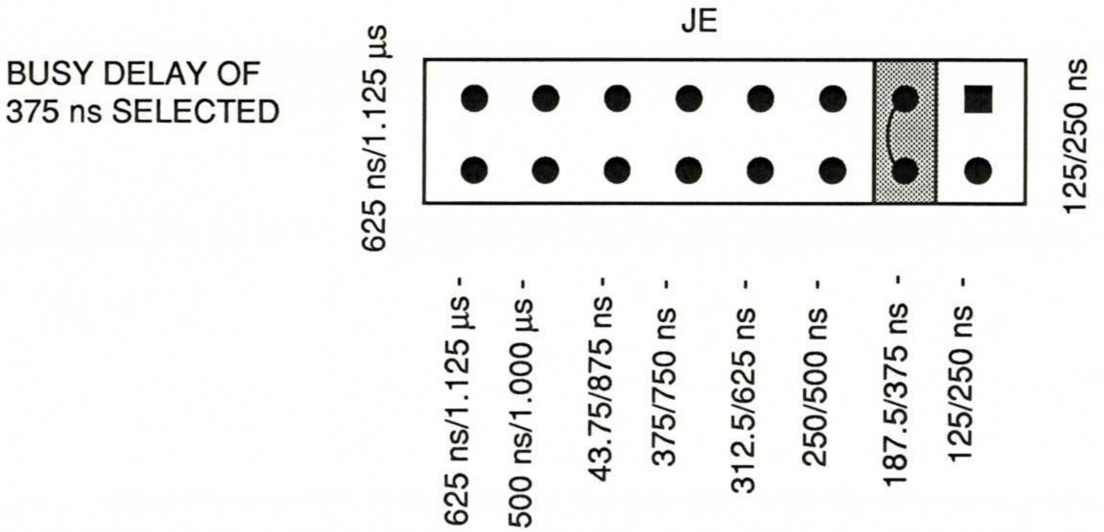
Figure 5.3.7-1. Busy Delay Clock Jumper

MDR11W-A/F5.3.7-1

5.3.8 Busy Delay

The BUSY DELAY jumper, JE, is an eight position jumper that allows the user to adjust the data skew delay on data output transfers. Depending on the Busy Delay Clock setting (see paragraph 5.3.7) chosen by jumper JK, the BUSY DELAY is adjustable in 62.5 ns intervals from 125 ns to 625 ns or in 125 ns intervals from 250 ns to 1125 ns. This delay time represents the minimum amount of delay added from presentation of output data from the Output Register until the BUSY output handshake signal is asserted. Figure 5.3.8-1 shows this jumper set with a 187.5/375 ns skew delay selected (the factory setting).

The longer duration delay range facilitates interfacing with user devices requiring relatively long data deskew times such as that required for the DEC DRV11W interface module. The shorter duration delay range accommodates the much shorter minimum data deskew delay requirement of the DEC DR11W as well as other user interfaces with relatively short data deskew delay time requirements. The 125 ns minimum BUSY DELAY can be useful in a controlled environment to maximize the data transfer rate between two VMIVME-DR11W-A Interface Boards in a back-to-back interface application.



MDR11W-A/F5.3.8-1

Figure 5.3.8-1. BUSY DELAY Jumper

5.4 ADDRESS MODIFIERS

The board is factory-configured via a programmed PAL to respond to either of four address modifier codes in the VMEbus slave short and standard I/O address space: short supervisory ($2D), and short nonprivileged access ($29), or standard supervisory access ($3D), and standard nonprivileged ($39). Figure 5.3-1 shows the location of the address modifier PAL. Figure 5.3.3-1 shows JH jumper used for selection of short I/O space or standard I/O space.

5.5 I/O CABLES

5.5.1 User Interface Cables

The VMIVME-DR11W-A provides a TTL compatible user interface that is terminated in 120 Ω via 180/390Ω terminating resistors. The VMIVME-DR11W-A may be connected to external devices, or to another VMIVME-DR11W-A using up to fifty (50) foot (15.2m) cables if the following guidelines are followed:

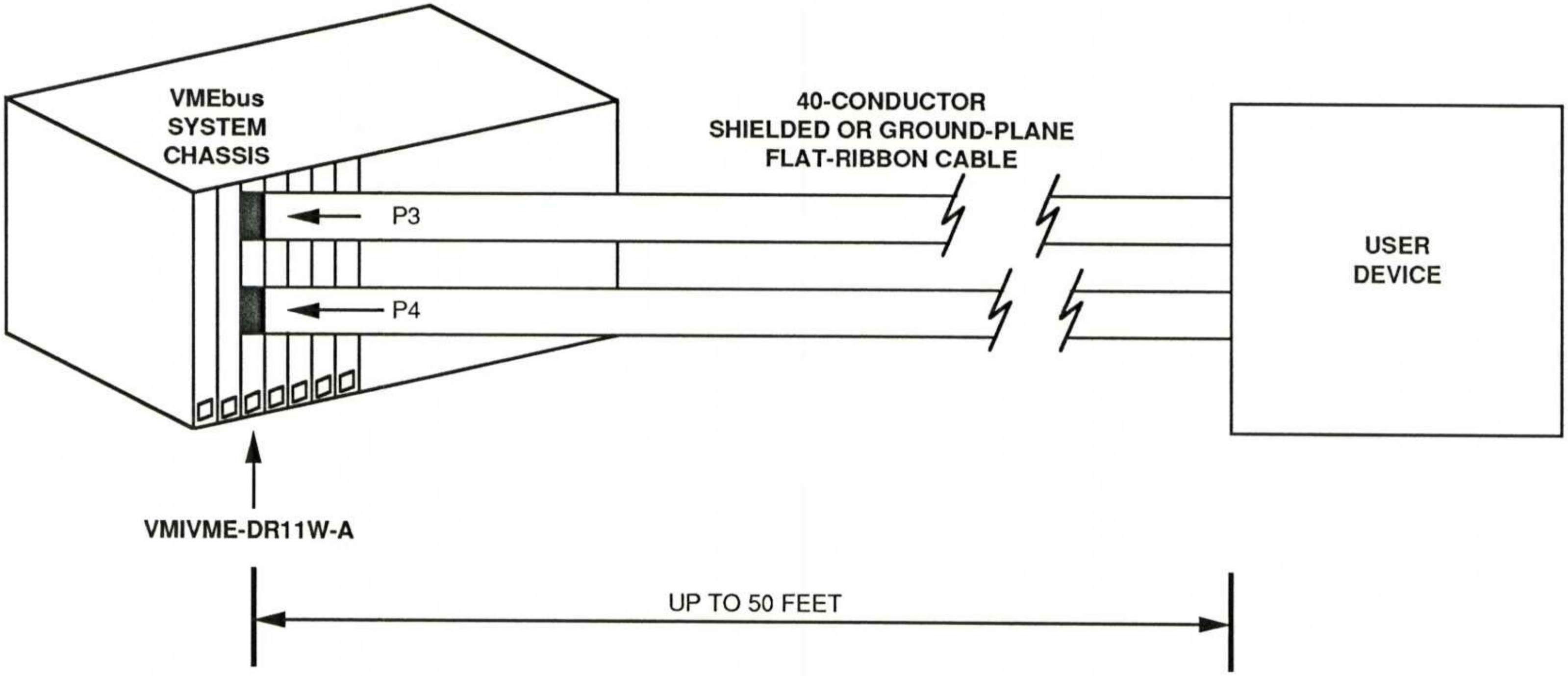
1. Shielded or ground-plane flat-cable (or superior cable) must be used. Mass-terminated unshielded flat-cable will cause data errors. A cable with individually shielded conductors or a signal/ground wire pair for each signal (40 pairs) should provide best results.
2. The cable shield must be grounded at both ends of the interface cable if a shielded cable is used. The cable grounds should have low impedance at high frequencies.
3. Shielded flat-cables typically have the shield connected to pin 1; therefore, pin 1 of the cable must connect to pin 40 of the connector (because the standard DR11W pin-out has data bit 15 tied to pin 1).
4. A special cable is required for the Macrolink interface. The interface on the Macrolink side requires that pin 1 on the connector connects to pin 1 of the cable and pin 40 of the connector connects to pin 40 of the cable.

Refer to Figures 5.5.1-1, 3.1-2, and 3.1-3 for proper cable connection to the user device. To connect two VMIVME-DR11W's back-to-back, P3 should be connected to P4 on the other device, and P4 should be connected to P3 on the other device.

The VMIVME-DR11W-A is pinout compatible with DR11-W and DRV-11B type devices. It is NOT pinout compatible with DR11-B type devices and damage may result if it is cabled directly to a DR11-B type device.

5.5.2 Macrolink Pin Number Reversal

The Macrolink DR11W compatible interface has pin 1 of the connector connected to ground and pin 40 connected to data bit 15. The special cable required can be ordered from VMIC.



MDR11W-A/F5.5.1-1

Figure 5.5.1-1. System Interconnection Pictorial Diagram

5.5.3 Pin Number Reversal Through Bulkhead Connector

If two flat cables are mated together by back-to-back male connectors, a pin number reversal will occur, i.e., the odd numbered conductors will be swapped with the even numbered conductors. Therefore, if the Digital Equipment Corporation DC06 cable kit is utilized, the short pin number reversing cable (in the kit) must be installed to correct the pin number reversal.

5.6 CABLE CONNECTOR

A compatible cable connector for the VMIVME-DR11W-A is Panduit No. 050-040-455. Strain relief Panduit No. is 100-000-058.

5.7 USER CONNECTOR PIN ASSIGNMENTS

User connector pin assignments and signal names are shown in Table 5.7-1.

5.8 INSTALLATION

The VMIVME-DR11W-A is compatible with any VMEbus chassis which can accept double Eurocard form factor boards with front panels. The board requires that the P2 backplane and memory be capable of supporting 32-bit transfers. Two captive screws are provided on the front panel to secure the board to the chassis, and handles are provided for removal and installation. A VMEbus CPU that is capable of supporting either 16-bit or 32-bit transfers to either standard or short I/O address space is desirable.

Table 5.7-1. Connector Pin Assignments

CONNECTOR P3

PIN	SIGNAL	PIN	SIGNAL
1 VV	D0 15 H	21 X	END CYCLE H
2 UU	D0 00 H	22 W	GND
3 TT	D0 14 H	23 V	STATUS C H
4 SS	D0 01 H	24 U	GND
5 RR	D0 13 H	25 T	STATUS C H
6 PP	D0 02 H	26 S	GND
7 NN	D0 12 H	27 R	STATUS B H
8 MM	D0 03 H	28 P	GND
9 LL	D0 11 H	29 N	INIT H
10 KK	D0 04 H	30 M	GND
11 JJ	D0 10 H	31 L	STATUS A H
12 HH	D0 05 H	32 K	BURST RQ L
13 FF	D0 09 H	33 J	WC INC ENB H*
14 EE	D0 06 H	34 H	GND
15 DD	D0 08 H	35 F	READY H
16 CC	D0 07 H	36 E	GND
17 BB	GND	37 D	ACLO FNCT 2 H
18 AA	GND	38 C	GND
19 Z	CYCLE RQB H	39 B	CYCLE RQA H
20 Y	GND	40A	GND

CONNECTOR P4

PIN	SIGNAL	PIN	SIGNAL
1 VV	DI 15 H	21 X	GO H
2 UU	DI 00 H	22 W	GND
3 TT	DI 14 H	23 V	FNCT 1 H
4 SS	DI 01 H	24 U	GND
5 RR	DI 13 H	25 T	C1 H
6 PP	DI 02 H	26 S	GND
7 NN	DI 12 H	27 R	FNCT 2 H
8 MM	DI 03 H	28 P	GND
9 LL	DI 11 H	29 N	CO H*
10 KK	DI 04 H	30 M	GND
11 JJ	DI 10 H	31 L	FNCT 3 H
12 HH	DI 05 H	32 K	FNCT 3 H
13 FF	DI 09 H	33 J	BA IND ENB H*
14 EE	DI 06 H	34 H	GND
15 DD	DI 08 H	35 F	AOO H***
16 CC	DI 07 H	36 E	GND
17 BB	GND	37 D	ATTN H
18 AA	GND	38 C	GND
19 Z	GND	39 B	BUSY**
20 Y	GND	40A	GND

*Not utilized by or necessary for operation of DR11W link.

**See Section 5.3.1 for polarity selection.

***Becomes "Other Board Ready" on-board.

MDR11W-A/T5.7-1

SECTION 6

MAINTENANCE AND WARRANTY

6.1 MAINTENANCE

This section of the technical manual provides information relative to the care and maintenance of VMIC's products. Should the products malfunction, the user should verify the following:

a. Software
b. System configuration
c. Electrical connections
d. Jumper or configuration options
e. Boards fully inserted into their proper connector location
f. Connector pins are clean and free from contamination
g. No components of adjacent boards are disturbed when inserting or removing the board from the VMEbus card cage
h. Quality of cables and I/O connections

User level repairs are not recommended. Contact VMIC for a Return Material Authorization (RMA) Number. This RMA Number must be obtained prior to any return.

6.2 MAINTENANCE PRINTS

The appendix(ices) to this manual contain(s) drawings and diagrams for reference purposes.

6.3 WARRANTY

VMIC's Standard Products are warranted to be free from defects in material and workmanship for a period of two years (24 months) from the date of shipment. In discharge of this warranty, VMIC, at its option, agrees to either repair or replace, at VMIC's facility and at VMIC's discretion, any part, component, subassembly accessory, or any hardware, software, or system product, which under proper and normal use proves defective in material and workmanship.

The customer shall provide notice to VMIC of each such defect within a reasonable time after the customer's discovery of such defect.

In order to return the defective product(s) or part(s), the customer must contact VMIC's Customer Service Department to obtain a Call Ticket Number. The defective product(s) or part(s) must also be properly boxed and weighed. After a VMIC Call Ticket Number and RMA Number have been obtained, the defective product(s) or part(s) may be returned (transportation collect for surface UPS) to VMIC. Any replaced or repaired product(s) or part(s) will be shipped back to the customer's at the expense of VMIC (also UPS surface).

The customer should be aware that the above process can sometimes take up to eight (8) days for the shipment to reach VMIC. The customer has the option to ship the defective product(s) or part(s) at the customer's own expense if the customer cannot afford this possible delay.

There shall be no warranty or liability on any VMIC product(s) or part(s) that is (are) damaged or subjected to accident(s), perils of nature, negligence, overtemperature, overvoltage, misapplication of electrical power, insertion or removal of boards from backplanes and/or I/O connectors with power applied by the customer(s), appointee(s), or any other person(s) without the expressed approval of VMIC.

Final determination of warranty eligibility shall be made by VMIC, and if a warranty claim is considered invalid for any reason, the customer will be charged for services performed and expenses incurred by VMIC in repair, handling and shipping the returned product or part. Determination as to whether the item is within warranty, coverage shall not be unreasonably withheld.

The warranty period of the replacement or repaired product(s) or part(s) shall terminate with the termination of the warranty period with respect to the original product(s) or part(s) for all replacement parts supplied or repairs made during the original warranty period.

THE FOREGOING WARRANTY AND REMEDY ARE EXCLUSIVE AND VMIC SHALL HAVE NO OTHER OR ADDITIONAL LIABILITY TO BUYER OR TO ANYONE CLAIMING UNDER BUYER (THIRD PARTY) UNDER ANY OTHER AGREEMENT OR WARRANTY, EXPRESS OR IMPLIED EITHER IN FACT OR BY OPERATION OF THE LAW, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS, STATUTORY, OR OTHERWISE. VMIC SHALL HAVE NO LIABILITY FOR SPECIAL OR CONSEQUENTIAL DAMAGES OF ANY KIND OR FROM ANY CAUSE ARISING OUT OF THE INSTALLATION OR USE OF ANY PRODUCT FURNISHED HEREUNDER.

6.4 OUT-OF-WARRANTY REPAIR POLICY

The following sections describe VMIC's policy on repairs and warranties on repaired products.

6.4.1 Repair Category

VMIC's repair policy of standard products is divided into two categories, depending on the item to be repaired. These categories are:

a. Product Exchange
b. Fixed Price Repair

Category 1 (product exchange) represents the fastest turn around of the two categories. In this case, the customer sends the malfunctioning product to VMIC. VMIC will return an operational product to the customer within 72 hours of receipt provided VMIC has the product in stock.

Provided that the returned product is repairable customers should contact VMIC prior to returning products for repair to determine stocking status.

Category 2 (Fixed Price Repair) applies to products returned to VMIC for repair and subsequent return to the customer.

Return authorizations are required on all product repairs, and all purchase orders should refer to VMIC's RMA Number which is assigned by VMIC's Customer Service Department.

6.4.2 Repair Pricing

Contact your factory representative for repair pricing. Current pricing can be found in the Repair and Replacement Policy in the most current Standard Conditions of Sales Document (F0109-91). Refer to exclusions (Section 6.4.7).

6.4.3 Payment

Payment is due upon delivery or at VMIC's option, net thirty (30) days from the date of delivery. Payment should be made to:

VME Microsystems International Corporation
12090 South Memorial Parkway
Huntsville, Alabama 35803-3308
Attention: Accounts Receivable

VMIC allows a one (1) percent discount for payment made within ten (10) days of invoice date or a two (2) percent discount on payment made prior to shipment of order. This payment discount, however, does not apply to freight.

6.4.4 Shipping Charges

Shipping charges are the customer's responsibility, with the exception of warranty repairs, whereby VMIC will pay the return to customer shipping charges.

6.4.5 Shipping Instructions

The type of packaging used to ship the product depends on whether the product is shipped singly, in a chassis, or packaged with other boards. The shipper should carefully pack the product(s), using the same precautions listed in the "unpacking procedures". The user should utilize the same (or equivalent) protective packaging container for re-shipment as provided by VMIC. Approved ESD procedures are recommended when handling VMIC's products.

6.4.6 Warranty on Repairs

Products repaired by VMIC are warranted against defects in workmanship and material for a period of ninety (90) days from date of shipment to the customer for all products that were repaired out of warranty. See Standard Conditions of Sale for products repaired within the warranty.

6.4.7 Exclusions

Repair rates may not apply to products which have received unusual physical or electrical damage. In such cases, VMIC will provide an estimated price for product repair or replacement. The customer may then choose to have the product repaired at the estimated price, returned unrepaired at no charge, or replaced at VMIC's current list price.

APPENDIX A

ASSEMBLY DRAWING, PARTS LIST, AND SCHEMATIC

APPENDIX B

INTEGRATED CIRCUIT SPECIFICATIONS

DESCRIPTION	**PART NO.**
Bus Interrupter Module	MC68153
Direct Memory Access Interface (DMAI)	SCB68430

MOTOROLA
SEMICONDUCTORS
P.O. BOX 20912 • PHOENIX, ARIZONA 85036

MC68153

Advance Information

TTL

BUS INTERRUPTER MODULE

ADVANCED LOW POWER SCHOTTKY

BUS INTERRUPTER MODULE

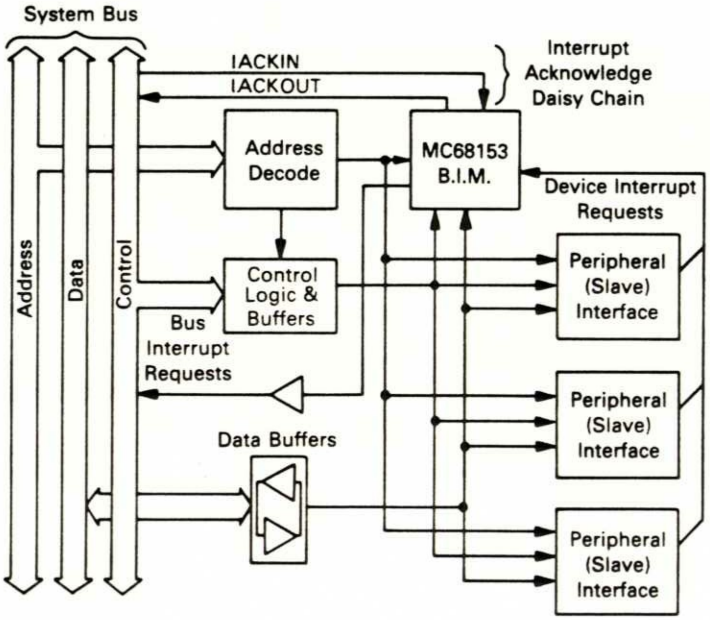
The bipolar LSI MC68153 Bus Interrupter interfaces a microcomputer system bus to multiple slave devices requiring interrupt capabilities. It handles up to 4 independent sources of interrupt requests and is fully programmable.

- VERSAbus/VMEbus Compatible
- MC68000 Compatible
- Handles 4 Independent Interrupt Sources
- 8 Programmable Read/Write Registers
- Programmable Interrupt Request Levels
- Programmable Interrupt Vectors
- Supports Interrupt Acknowledge Daisy Chain
- Control Registers Contain Flag Bits
- Single +5.0 Volt Supply
- Total Power Dissipation = 1.5 W Typical
- Temperature Range of 0°C to 70°C
- Chip Access Time = 200 ns Typical with 16 MHz Clock
- 40-Pin Dual-In-Line Package

P SUFFIX
PLASTIC PACKAGE
CASE 711-03

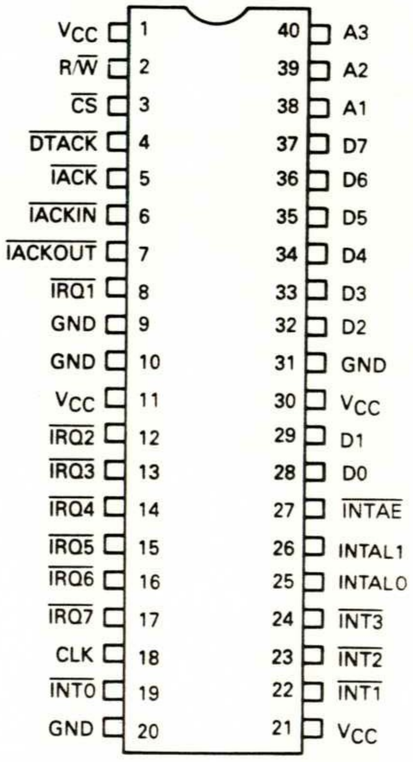
L SUFFIX
CERAMIC PACKAGE
CASE 734-04

FIGURE 1 — MC68153 SYSTEM BLOCK DIAGRAM


System Bus
IACKIN
IACKOUT
Interrupt Acknowledge Daisy Chain
Address Decode
MC68153 B.I.M.
Device Interrupt Requests
Address
Data
Control
Control Logic & Buffers
Peripheral (Slave) Interface
Bus Interrupt Requests
Peripheral (Slave) Interface
Data Buffers
Peripheral (Slave) Interface

VERSAbus is a trademark of Motorola.

PIN ASSIGNMENTS



This document contains information on a new product. Specifications and information herein are subject to change without notice.

ABSOLUTE MAXIMUM RATINGS (Beyond which useful life may be impaired.)

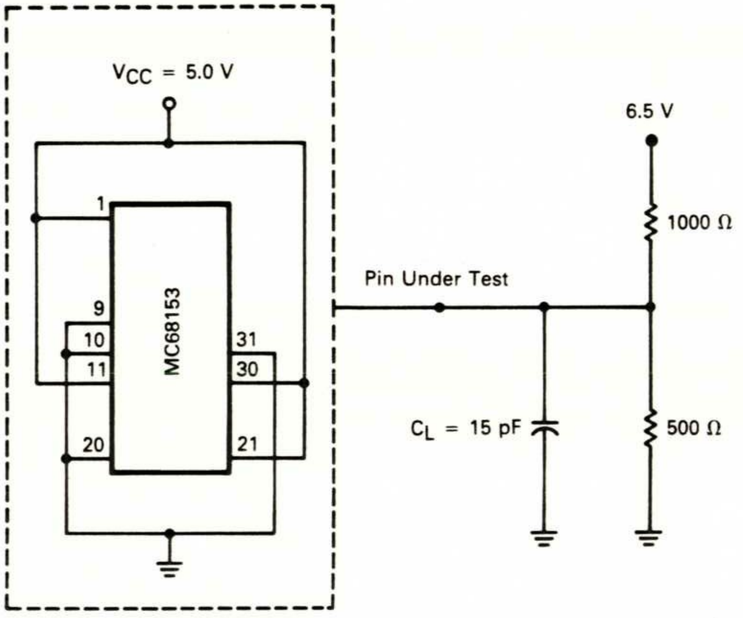
Parameter	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.5 to +7.0	V
Input Voltage	V_{in}	−0.5 to +7.0	V
Input Current	I_{in}	−30 to +5.0	mA
Output Voltage	V_{out}	−0.5 to +5.5	V
Output Current	I_{OL}	Twice Rated I_{OL}	mA
Storage Temperature	T_{stg}	−65 to +140	°C
Junction Operating Temperature	T_J	−55 to +140	°C

BURN-IN LIMITS: A maximum T_J of +175°C may be used for periods not to exceed 250 hours.

DC ELECTRICAL SPECIFICATIONS (V_{CC} = 5.0 V ±5%, T_A = 0°C to 70°C)

Parameter	Symbol	Min	Max	Unit	Test Conditions
High Level Input Voltage	V_{IH}	2.0	—	V	
Low Level Input Voltage	V_{IL}	—	0.8	V	
Input Clamp Voltage	V_{IK}	—	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
High Level Output Voltage[1]	V_{OH}	2.7	—	V	V_{CC} = MIN, I_{OH} = −400 μA
Low Level Output Voltage	V_{OL}	—	0.4	V	V_{CC} = MIN, I_{OL} = 8.0 mA
Output Short Circuit Current[2]	I_{OS}	−15	−130	mA	V_{CC} = MAX, V_{OUT} = 0 V
High Level Input Current	I_{IH}	—	20	μA	V_{CC} = MAX, V_{IN} = 2.7 V
Low Level Input Current	I_{IL}	—	−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V
Supply Current	I_{CC}	225	385	mA	V_{CC} = MAX
Output Off Current (High)	I_{OZH}	—	20	μA	V_{CC} = MAX, V_{OUT} = 2.4 V
Output Off Current (Low)	I_{OZL}	—	−20	μA	V_{CC} = MAX, V_{OUT} = 0.4 V

AC TEST CIRCUIT — AC Testing of All Outputs



NOTES:

1. Not applicable to open-collector outputs.
2. Not more than one output should be shorted at a time for longer than one second.
3. $\overline{CS}$ Low to CLK High (Setup Time) of 15 ns Min must be observed.
4. $\overline{IACK}$ Low to CLK High and $\overline{IACKIN}$ Low to CLK High (Setup Times) of 15 ns Min must be observed.
5. See Table 1 for additional performance guidelines.

AC ELECTRICAL SPECIFICATIONS (V_{CC} = 5.0 V ±5%, T_A = 0°C to 70°C)

Parameter	Test Number[5]	Max (ns)
CLK High to Data Out Valid (Delay)[3]	1	55
CLK High to $\overline{\text{DTACK}}$ Low (Delay)[3]	2	40
$\overline{\text{CS}}$ High to $\overline{\text{DTACK}}$ High (Delay)	3	35
CLK High to Data Out Valid (Delay)[4]	4	55
CLK High to $\overline{\text{INTAE}}$ Low (Delay)[4]	5	40
$\overline{\text{IACK}}$ High to Data Out High Impedance (Delay)	6	60
$\overline{\text{IACK}}$ High to $\overline{\text{DTACK}}$ High (Delay)	7	45
$\overline{\text{CS}}$ High to Data Out High (Delay)	8	45
CS High to IRQ High (Delay)	9	60
$\overline{\text{IACK}}$ High to $\overline{\text{INTAE}}$ High (Delay)	10	35

GENERAL DESCRIPTION

The MC68153 Bus Interrupter Module (BIM) is designed to serve as an interrupt requester for peripheral devices in a microcomputer system. Up to 4 independent devices can be interfaced to the system bus by the MC68153. Intended for asynchronous master/slave bus operation, the BIM is compatible with VERSAbus, VMEbus, MC68000 device bus, and other system buses. Figure 1 shows a block diagram of a typical configuration. In this example, three peripheral devices (bus slaves) are connected to the system data bus. Each of these devices could be parallel I/O, serial I/O, or some other function. An interrupt request from any device is routed to the MC68153, and the BIM handles all interface to the system bus. It generates a bus interrupt request as a result of the device interrupt request. When the system interrupt handler or processor responds with an interrupt acknowledge cycle, the MC68153 can answer supplying an interrupt vector and handling all timing.

The functional block diagram of the MC68153 is shown in Figure 2. The device contains circuitry to accept four separate interrupt sources ($\overline{\text{INT0}}$ – $\overline{\text{INT3}}$). Interface to the system bus includes generation of bus interrupt requests ($\overline{\text{IRQ1}}$ – $\overline{\text{IRQ7}}$), response to a bus interrupt acknowledge cycle (either supplying a vector or passing on a daisy chain signal), and releasing the bus interrupt request signal at the proper time. The BIM has flexibility provided by eight programmable read/write registers. Four 8-bit vector registers (VR0 – VR3) contain status/address information and supply a byte vector in response to an interrupt acknowledge cycle for the corresponding interrupt source. Four other 8-bit control registers (CR0 – CR3) contain information that oversees operation of the interrupt circuitry. The control information is programmable and includes interrupt request level and interrupt enable and disable. Also contained in the control registers are flag-bits. These flags are useful for task coordination, resource management, and interprocessor communication.

SIGNAL DESCRIPTION

Throughout the data sheet, signals are presented using the terms asserted and negated independent of whether the signal is asserted in the high voltage or low voltage state. Active low signals are denoted by a superscript bar.

BIDIRECTIONAL DATA BUS — D0 – D7

Pins D0 – D7 form an 8-bit bidirectional data bus to/from the system bus. These are active high, 3-state pins. D7 is the most significant bit.

ADDRESS INPUTS — A1 – A3

These active high inputs serve two functions. One function is to select one of the eight possible registers during a read or write cycle. Secondly, during an interrupt acknowledge A1 – A3 show the level of interrupt being acknowledged, and the BIM uses these to determine if a match exists with an internal level.

CHIP SELECT — $\overline{\text{CS}}$

$\overline{\text{CS}}$ is an active low input used to select the BIM's registers for the current bus cycle. Address strobe, data strobe, and appropriate address bits must be included in the chip select equation.

READ/WRITE — R/$\overline{\text{W}}$

The R/$\overline{\text{W}}$ input is a signal from the system bus used to determine if the current bus cycle is a read (high) or write (low).

DATA TRANSFER ACKNOWLEDGE — $\overline{\text{DTACK}}$

$\overline{\text{DTACK}}$ is an open-collector, active low output that signals the completion of a read, write, or interrupt acknowledge cycle. During read or interrupt acknowledge cycles, $\overline{\text{DTACK}}$ is asserted by the MC68153 after data has been provided on the data bus; during write cycles it is asserted after data has been accepted from the data bus. A pullup resistor is required to maintain $\overline{\text{DTACK}}$ high between bus cycles.

FIGURE 2 — MC68153 FUNCTIONAL BLOCK DIAGRAM

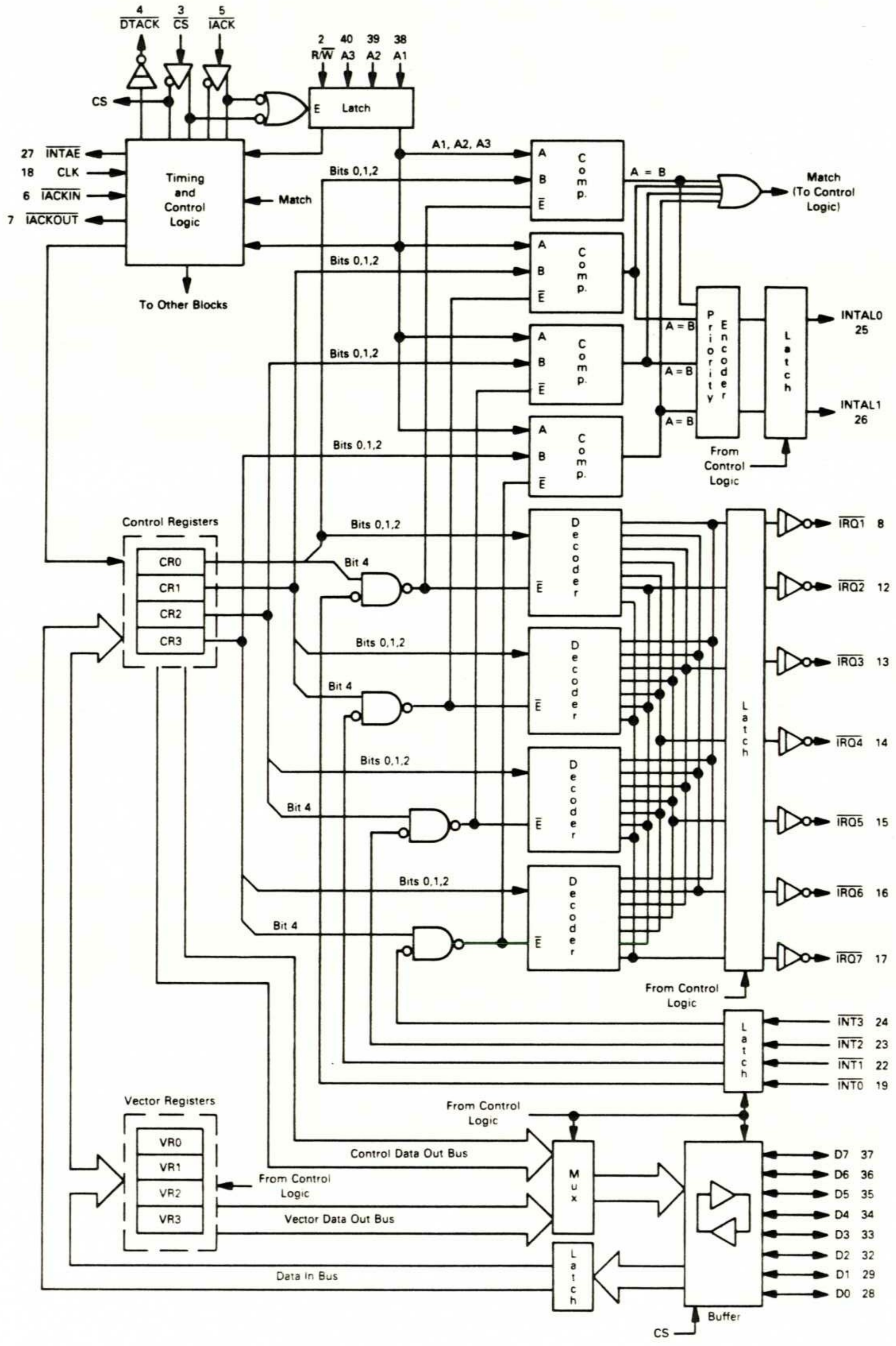
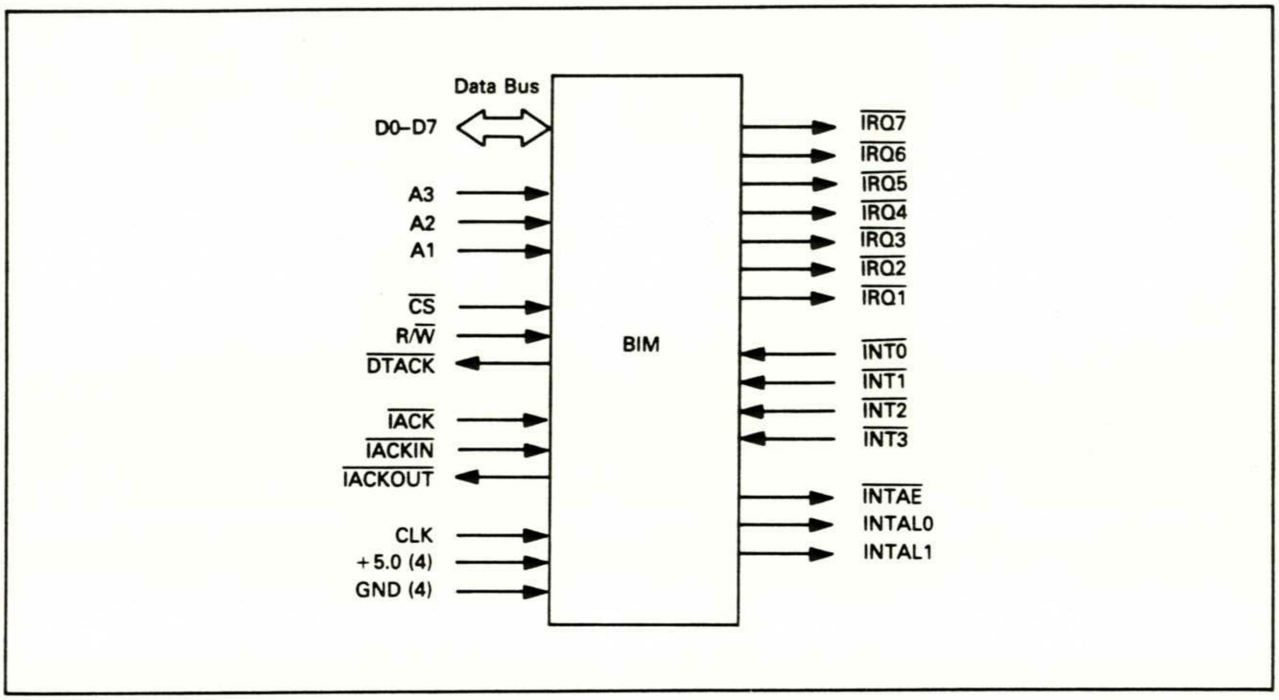


FIGURE 3 — LOGICAL PIN ASSIGNMENT



INTERRUPT ACKNOWLEDGE SIGNALS — $\overline{\text{IACK}}$, $\overline{\text{IACKIN}}$, $\overline{\text{IACKOUT}}$

These three pins support the interrupt acknowledge cycle. A low level on the $\overline{\text{IACK}}$ input indicates an interrupt acknowledge cycle has been initiated. This signal is conditioned externally with Address Strobe and the lower data strobe of an MC68000 type bus. After $\overline{\text{IACK}}$ is asserted the BIM compares the interrupt level presented on address lines A1, A2, and A3 with the current levels generated internally and determines if a match exists. Then, if input $\overline{\text{IACKIN}}$ is asserted (driven low), the BIM will either complete the interrupt acknowledge cycle if a match exists or assert output $\overline{\text{IACKOUT}}$ if *no* match exists.

$\overline{\text{IACKIN}}$ and $\overline{\text{IACKOUT}}$ form part of a prioritized interrupt acknowledge daisy chain. The daisy chain prioritizes interrupters and guarantees that two or more devices requesting an interrupt on the same level will not respond to the same cycle. The requesting device (or interrupter) must wait until $\overline{\text{IACKIN}}$ is asserted and not pass the signal on (assert $\overline{\text{IACKOUT}}$) if it is to complete the interrupt acknowledge cycle.

BUS INTERRUPT REQUEST SIGNALS — $\overline{\text{IRQ1}}$ - $\overline{\text{IRQ7}}$

These open-collector outputs are low when asserted, indicating a bus interrupt is requested at the corresponding level. An open-collector buffer is normally required for sufficient drive when interfacing to a system bus. A pullup resistor is required to maintain $\overline{\text{IRQ1}}$ – $\overline{\text{IRQ7}}$ high between interrupt requests.

DEVICE INTERRUPT REQUEST SIGNALS — $\overline{\text{INT0}}$ - $\overline{\text{INT3}}$

$\overline{\text{INT0}}$ – $\overline{\text{INT3}}$ are active low inputs used to indicate to the BIM that a device wants a bus interrupt.

INTERRUPT ACKNOWLEDGE ENABLE — $\overline{\text{INTAE}}$

During an interrupt acknowledge cycle, this output pin is asserted low to indicate that outputs INTAL0 and INTAL1 are valid. These two outputs contain an encoded number (x) corresponding to the interrupt ($\overline{\text{INTx}}$) being acknowledged. This feature can be used to signal interrupting devices, which supply their own vector, when to respond to the interrupt acknowledge cycle with the vector and a $\overline{\text{DTACK}}$ signal.

INTERRUPT ACKNOWLEDGE LEVEL — INTAL0, INTAL1

These active high outputs contain an encoded number corresponding to the interrupt level being acknowledged. They are valid only when $\overline{\text{INTAE}}$ is asserted low.

CLOCK — CLK

The CLK input is used to supply the clock for internal operations of the MC68153.

RESET — $\overline{\text{CS}}$, $\overline{\text{IACK}}$

Although a reset input is not supplied, an on-board reset is performed if $\overline{\text{CS}}$ and $\overline{\text{IACK}}$ are asserted simultaneously.

FIGURE 4 — MC68153 REGISTER MODEL

ADDRESS BIT A3	A2	A1	FLAG	FLAG AUTO-CLEAR	EXTERNAL/$\overline{\text{INTERNAL}}$	INTERRUPT ENABLE	INTERRUPT AUTO-CLEAR	INTERRUPT LEVEL			
0	0	0	F	FAC	X/$\overline{\text{IN}}$	IRE	IRAC	L2	L1	L0	CONTROL REGISTER 0
0	0	1	F	FAC	X/$\overline{\text{IN}}$	IRE	IRAC	L2	L1	L0	CONTROL REGISTER 1
0	1	0	F	FAC	X/$\overline{\text{IN}}$	IRE	IRAC	L2	L1	L0	CONTROL REGISTER 2
1	1	1	F	FAC	X/$\overline{\text{IN}}$	IRE	IRAC	L2	L1	L0	CONTROL REGISTER 3
1	0	0	V7	V6	V5	V4	V3	V2	V1	V0	VECTOR REGISTER 0
1	0	1	V7	V6	V5	V4	V3	V2	V1	V0	VECTOR REGISTER 1
1	1	0	V7	V6	V5	V4	V3	V2	V1	V0	VECTOR REGISTER 2
1	1	1	V7	V6	V5	V4	V3	V2	V1	V0	VECTOR REGISTER 3
			7	6	5	4	3	2	1	0	REGISTER NAME
			REGISTER BIT								

REGISTER DESCRIPTION

The MC68153 contains 8 programmable read/write registers. There are four control registers (CR0 – CR3) that govern operation of the device. The other four (VR0 – VR3) are vector registers that contain the vector data used during an interrupt acknowledge cycle. Figure 4 illustrates the device register model.

CONTROL REGISTERS

There is a control register for each interrupt source, i.e., CR0 controls $\overline{\text{INT0}}$, CR1 controls $\overline{\text{INT1}}$, etc. The control registers are divided into several fields:

1. Interrupt level (L2, L1, L0) — The least significant 3-bit field of the register determines the level at which an interrupt will be generated:

L2	L1	L0	IRQ LEVEL
0	0	0	DISABLED
0	0	1	IRQ1
0	1	0	IRQ2
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

A value of zero in the field disables the interrupt.

2. Interrupt Enable (IRE) — This field (Bit 4) must be set (high level) to enable the bus interrupt request associated with the control register. Thus, if the $\overline{\text{INTX}}$ line is asserted and IRE is cleared, *no* interrupt request ($\overline{\text{IRQX}}$) will be asserted.
3. Interrupt Auto-Clear (IRAC) — If the IRAC is set (Bit 3), IRE (Bit 4) is cleared during an interrupt acknowledge cycle responding to this request. This action of clearing IRE disables the interrupt request. To re-enable the interrupt associated with this register, IRE must be set again by writing to the control register.
4. External/Internal (X/$\overline{\text{IN}}$) — Bit 5 of the control register determines the response of the MC68153 during an interrupt acknowledge cycle. If the X/$\overline{\text{IN}}$ bit is clear (low level) the BIM will respond with vector data and a $\overline{\text{DTACK}}$ signal, i.e., an internal response. If X/$\overline{\text{IN}}$ is set, the vector is not supplied and no $\overline{\text{DTACK}}$ is given by the BIM, i.e., an external device should respond.
5. Flag (F) — Bit 7 is a flag that can be used in conjunction with the test and set instruction of the MC68000. It can be changed without affecting chip operation. It is useful for processor-to-processor communication and resource allocation.
6. Flag Auto-Clear (FAC) — If FAC (Bit 6) is set, the Flag bit is automatically cleared during an interrupt acknowledge cycle.

VECTOR REGISTERS

Each interrupt input has its own associated vector register. Each register is 8 bits wide and supplies a data byte during its interrupt acknowledge cycle if the associated External/Internal (X/$\overline{\text{IN}}$) control register bit is clear (zero). This data can be status, identification, or address information depending on system usage. The information is programmed by the system user.

DEVICE RESET

When the MC68153 is reset, the registers are set to a known condition. The control registers are set to all zeros (low). The vector registers are set to \$0F. This value is the MC68000 vector for an uninitialized interrupt vector.

FUNCTIONAL DESCRIPTION

SYSTEM OVERVIEW

The MC68153 can be used with many system buses, however, it is primarily intended for VMEbus, VERSAbus and MC68000 applications. Figure 5 shows a system configuration similar to VMEbus. In the figure only one system Data Transfer Bus (DTB) master is used. The Priority Interrupt structure provides a means for peripheral slave devices to ask for an interrupt of other processor (DTB master) activity and receive service from the processor. The MC68153 BIM acts as an interface device requesting and responding to interrupt acknowledge cycles for up to 4 independent slaves.

In Figure 5, functional modules are identified as Interrupters and an Interrupt Handler. An Interrupter (such as the MC68153) receives slave requests for an interrupt and handles all interface to the system bus required to ask for and respond to interrupt requests. The Interrupt Handler receives the bus interrupt requests, determines when an interrupt acknowledge will occur and at which level, and finally either performs the interrupt acknowledge (IACK) cycle or tells the DTB master to execute the IACK cycle.

The signal lines in the Priority Interrupt structure include (* — indicates active low):

1. IRQ1*-IRQ7* — seven prioritized interrupt request lines.
2. IACK* — signal line that indicates an interrupt acknowledge cycle is occurring.
3. IACKIN*/IACKOUT* — two signals that form part of a daisy chain that prioritizes interrupters.

In addition Data Transfer Bus control signals are involved in the IACK bus cycle:

1. AS* — the Address Strobe asserted low indicates a valid address is on the bus.
2. DSO* — the lower Data Strobe asserted low indicates a data transfer will occur on bus bits D00–D07.
3. WRITE* — the Read/Write is negated indicating the data is to be read from the Interrupter.
4. A01–A03 — Address lines A01–A03 contain the encoded priority level of the IACK cycle.
5. D00–D07 — Data bus lines D00–D07 are used to pass the interrupt vector from the responding Interrupter to the Interrupt Handler.
6. DTACK* — Data Transfer Acknowledge asserted low signals that the Interrupter has put the vector on the data bus.

FIGURE 5 — SIMPLE VMEbus CONFIGURATION

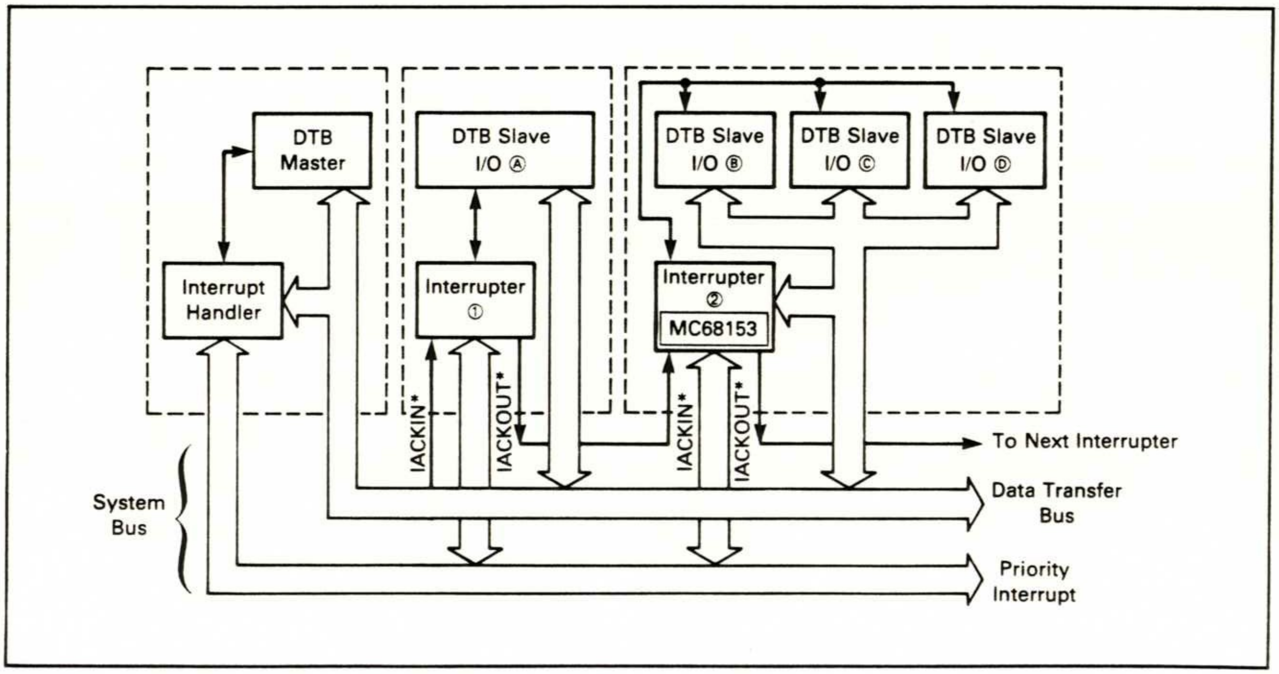
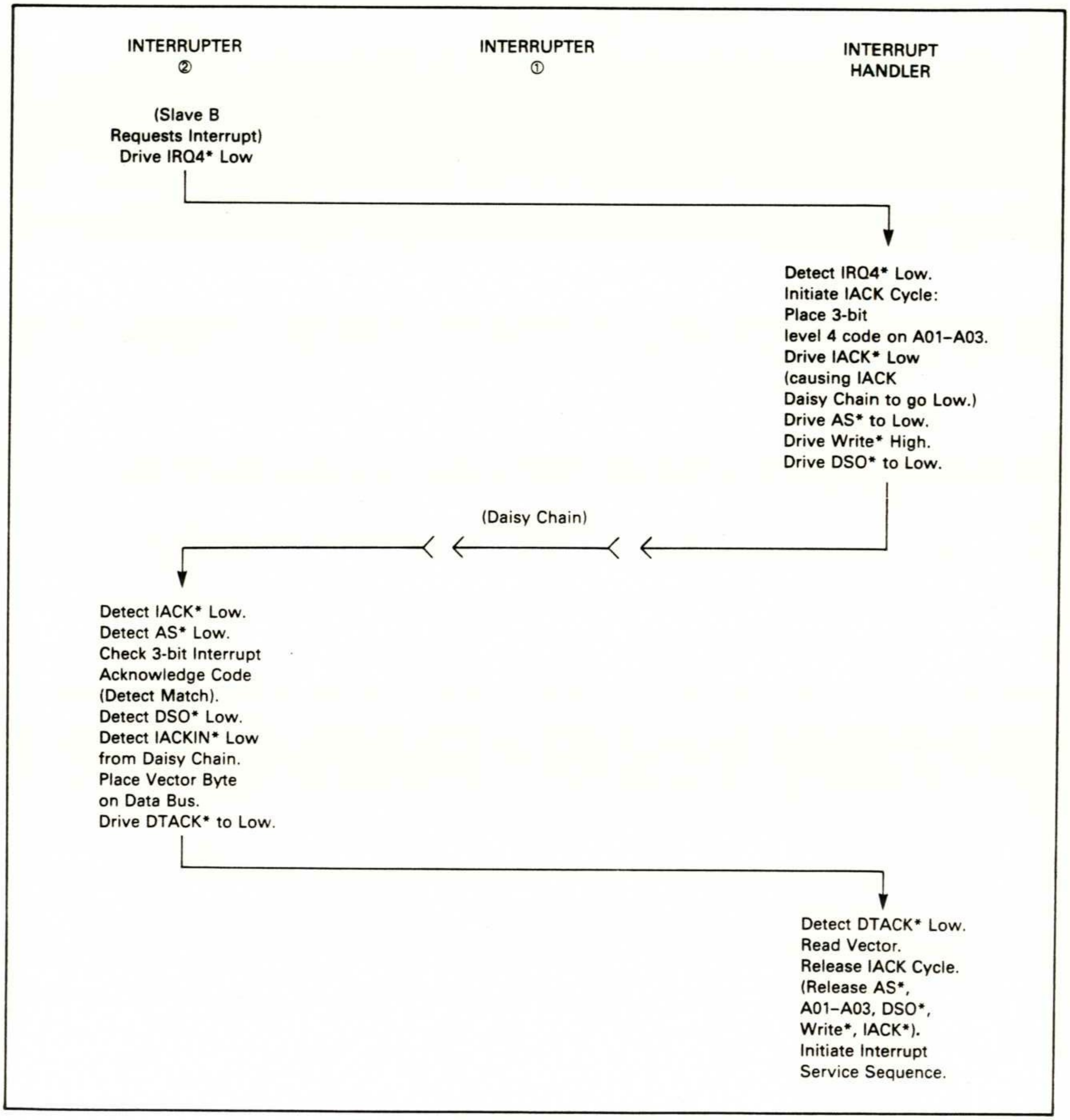


Figure 6 shows a flow diagram of a typical interrupt request and acknowledge operation. Briefly, the sequence of events is first, an Interrupter makes a request, next the Handler responds with an IACK cycle, then the Interrupter passes a vector to the Handler completing the IACK cycle, and finally the Handler uses the vector to determine additional action. Typically, an interrupt service routine is stored in software and the vector determines where its starting address is stored.

Note the daisy chain operation. If the IACK level (on A01–A03) does not match the Interrupter's request level or if no request is pending, the Interrupter passes the IACKIN* signal on and asserts IACKOUT*. This sequential action automatically prioritizes Requesters on the same level (first one in line with a request pending gets serviced) and prevents two or more Interrupters from responding simultaneously.

FIGURE 6 — INTERRUPT REQUEST AND ACKNOWLEDGE OPERATION FLOW DIAGRAM


INTERRUPTER ②
INTERRUPTER ①
INTERRUPT HANDLER
(Slave B Requests Interrupt) Drive IRQ4* Low
Detect IRQ4* Low. Initiate IACK Cycle: Place 3-bit level 4 code on A01–A03. Drive IACK* Low (causing IACK Daisy Chain to go Low.) Drive AS* to Low. Drive Write* High. Drive DSO* to Low.
(Daisy Chain)
Detect IACK* Low. Detect AS* Low. Check 3-bit Interrupt Acknowledge Code (Detect Match). Detect DSO* Low. Detect IACKIN* Low from Daisy Chain. Place Vector Byte on Data Bus. Drive DTACK* to Low.
Detect DTACK* Low. Read Vector. Release IACK Cycle. (Release AS*, A01–A03, DSO*, Write*, IACK*). Initiate Interrupt Service Sequence.

This discussion is a very cursory look at the bus operation. For more details including situations with multiple bus masters, the user is directed to the VMEbus Specification MVMEBS or VERSAbus Specification M68KVBS. Also, the MC68153 can be used with other buses having similar interrupt structures.

BIM BUS INTERFACE

Figure 7 shows a simplified block diagram of the MC68153 interface to VERSAbus or VMEbus. Address Decode and Control Logic are dependent on the application and must be designed to guarantee BIM ac specifications. It is possible in most cases that the decode logic can be shared with the slave devices. Buffers are provided where shown to comply with bus loading and drive specifications. It is also possible that buffers can be shared with the slave bus interface.

READ/WRITE OPERATION

All eight BIM registers can be accessed from the system bus in both read and write modes. The BIM has an asynchronous bus interface, primarily designed for MC68000-like buses. The following BIM signals generate read and write cycles: Chip Select ($\overline{\text{CS}}$), Read/Write (R/$\overline{\text{W}}$), Address Inputs (A1–A3), Data Bus (D0–D7), and Data Transfer Acknowledge ($\overline{\text{DTACK}}$). During read and write cycles the internal registers are selected by A1, A2, and A3 in compliance with the Figure 4 Truth Table.

Figure 8 shows the device timing for a read cycle. R/$\overline{\text{W}}$ and A1–A3 are latched on the falling edge of $\overline{\text{CS}}$ and must meet specified setup and hold times. Chip access time for valid data and $\overline{\text{DTACK}}$ are dependent on the clock frequency as shown in the figure.

Figure 9 shows the device timing for a write cycle. R/$\overline{\text{W}}$, A1–A3, and D0–D7 are latched on the falling edge of $\overline{\text{CS}}$ and must meet specified setup and hold times. Chip access time for $\overline{\text{DTACK}}$ is dependent on the clock frequency as shown in the figure.

FIGURE 7 — VMEbus/VERSAbus INTERFACE BLOCK DIAGRAM

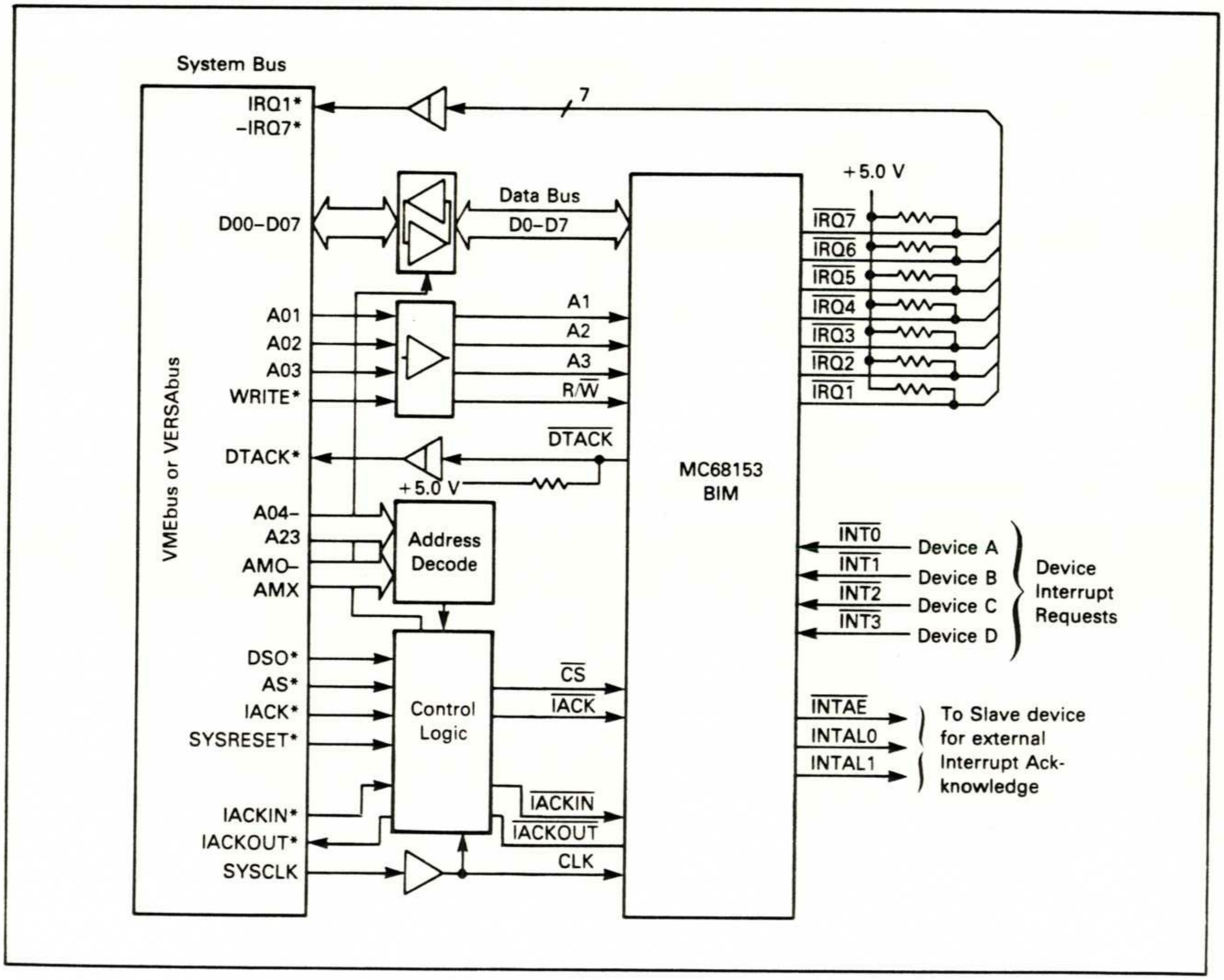


FIGURE 8 — READ CYCLE

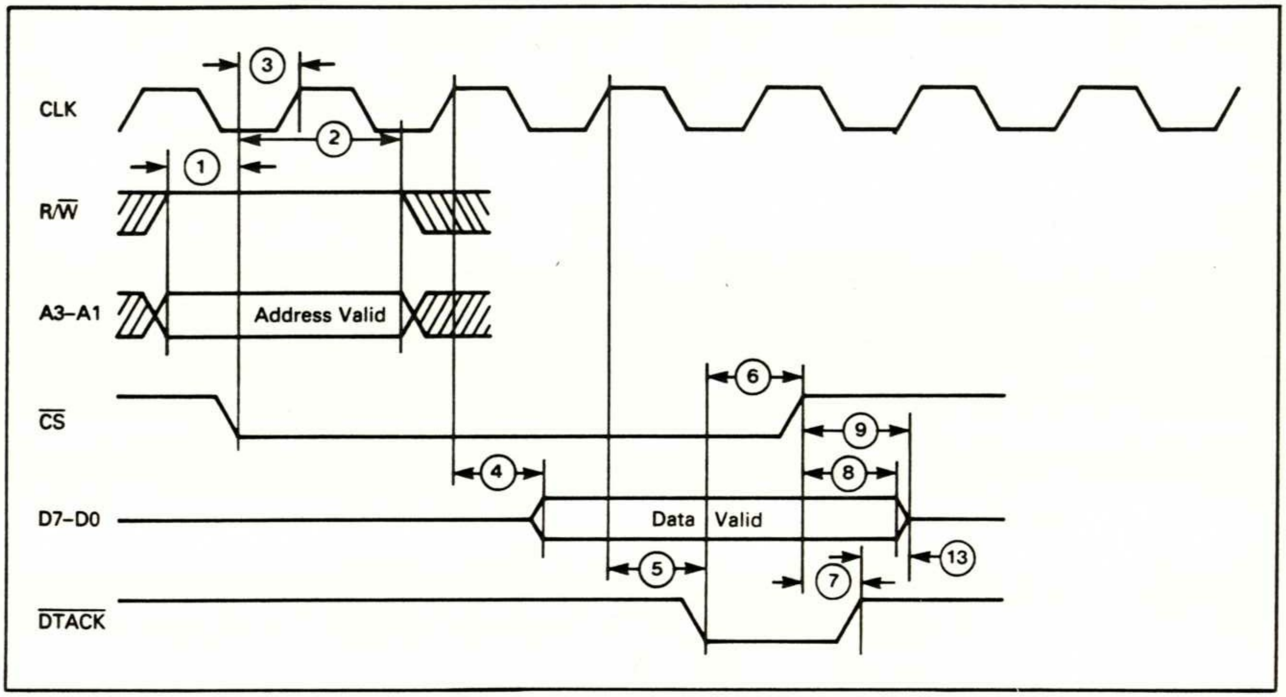
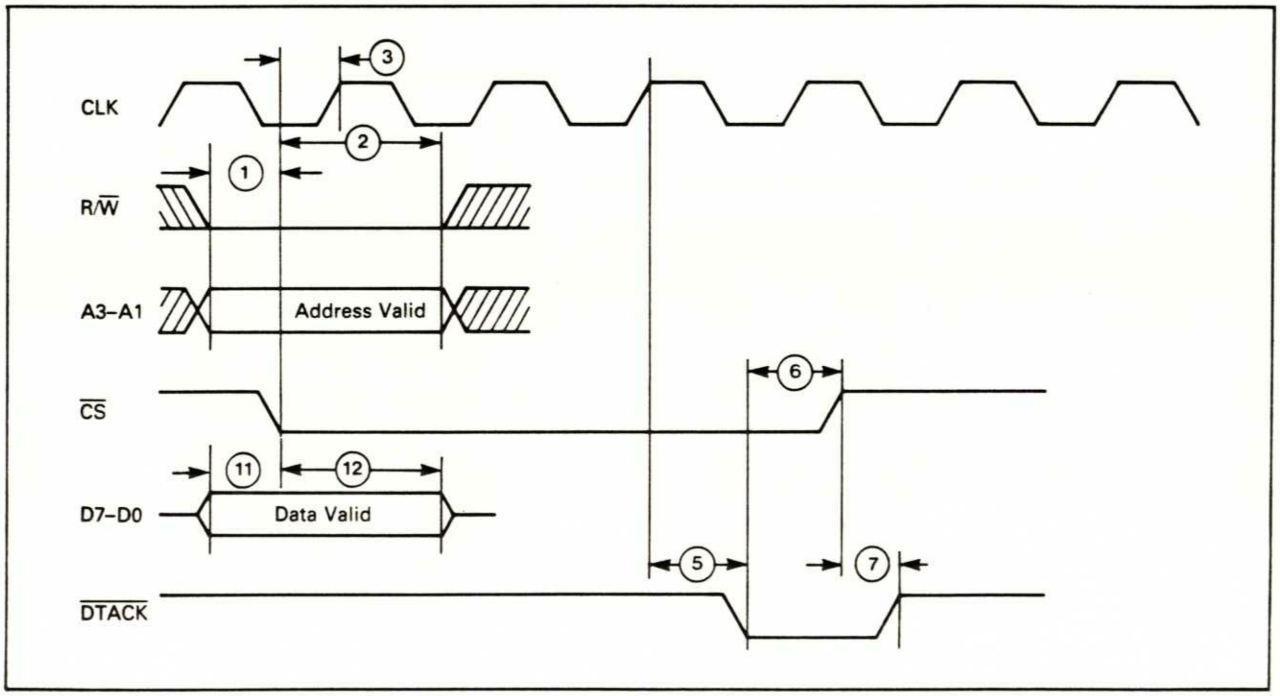


FIGURE 9 — WRITE CYCLE



INTERRUPT REQUESTS

The MC68153 accepts device interrupt requests on inputs $\overline{\text{INT0}}$, $\overline{\text{INT1}}$, $\overline{\text{INT2}}$, and $\overline{\text{INT3}}$. Each input is regulated by Bit 4 (IRE) of the associated control register (CR0 controls $\overline{\text{INT0}}$, CR1 controls $\overline{\text{INT1}}$, etc). If IRE (Interrupt Enable) is set and a device input is asserted, an Interrupt Request open-collector output ($\overline{\text{IRQ1}}$–$\overline{\text{IRQ7}}$) is asserted. The asserted $\overline{\text{IRQX}}$ output is selected by the value programmed in Bits 0, 1, and 2 of the control register (L0, L1, and L2). This 3-bit field determines the interrupt request level as set by software.

Two or more interrupt sources can be programmed to the same request level. The corresponding $\overline{\text{IRQX}}$ output will remain asserted until multiple interrupt acknowledge cycles respond to all requests.

If the interrupt request level is set to zero, the interrupt is disabled because there is no corresponding IRQ output.

INTERRUPT ACKNOWLEDGE

The response of an Interrupt Handler to a bus interrupt request is an interrupt acknowledge cycle. The IACK cycle is initiated in the MC68153 by receiving $\overline{\text{IACK}}$ low. R/$\overline{\text{W}}$, A1, A2, A3 are latched, and the interrupt level on line A1–A3 is compared with any interrupt requests pending in the chip. Further activity can be one of four cases:

1. No further action required — This occurs if $\overline{\text{IACKIN}}$ is **not** asserted. Asserting $\overline{\text{IACK}}$ only starts the BIM activity. If the daisy chain signal never reaches the MC68153 ($\overline{\text{IACKIN}}$ is not asserted), another Interrupter has responded to the $\overline{\text{IACK}}$ cycle. The cycle will end, the chip $\overline{\text{IACK}}$ is negated, and no additional action is required.

2. Pass on the interrupt acknowledge daisy chain — For this case, $\overline{\text{IACKIN}}$ input is asserted by the preceding daisy chain Interrupter, and $\overline{\text{IACKOUT}}$ output is in turn asserted. The daisy chain signal is passed on when no interrupts are pending on a matching level or when any possible interrupts are disabled. The Interrupt Enable (IRE) bit of a control register can disable any interrupt requests, and in turn, any possible matches.

3. Respond internally — For this case, $\overline{\text{IACKIN}}$ is asserted and a match is found. The MC68153 completes the IACK cycle by supplying an interrupt vector from the proper vector register followed by a $\overline{\text{DTACK}}$ signal asserted. $\overline{\text{IACKOUT}}$ is not asserted because the interrupt acknowledge cycle is completed by this device.

   For the MC68153 to respond in this mode of operation, the EXTERNAL/$\overline{\text{INTERNAL}}$ control register bit (X/$\overline{\text{IN}}$) must be zero. For each source of interrupt request, the associated control register determines the BIM response to an IACK cycle, and the X/$\overline{\text{IN}}$ bit sets this response either internally (X/$\overline{\text{IN}}$ = 0) or externally (X/$\overline{\text{IN}}$ = 1).

4. Respond externally — For the final case, $\overline{\text{IACKIN}}$ is also asserted, a match is found and the associated control register has X/$\overline{\text{IN}}$ bit set to one. The MC68153 does not assert $\overline{\text{IACKOUT}}$ and does assert $\overline{\text{INTAE}}$ low. $\overline{\text{INTAE}}$ signals that the requesting device must complete the IACK cycle (supplying a vector and $\overline{\text{DTACK}}$) and that the 2-bit code contained on outputs INTAL0 and INTAL1 shows which interrupt source is being acknowledged.

These cases are discussed in more detail in the following paragraphs.

Internal Interrupt Acknowledge

For an internal interrupt acknowledge to occur, the following conditions must be met:

1. One or more device interrupt inputs ($\overline{\text{INT0}}$–$\overline{\text{INT3}}$) has been asserted and corresponding control bit IRE value is one.

2. $\overline{\text{IACK}}$ asserted.

3. A match exists between [A3, A2, A1] and the [L2, L1, L0] field of an enabled, requesting control register. If two or more devices are requesting at the same interrupt level, preference is given to the highest number requester, that is, $\overline{\text{INT3}}$ has highest priority and $\overline{\text{INT0}}$ has lowest.

4. Control register bit X/$\overline{\text{IN}}$ of matching interrupt source must be zero.

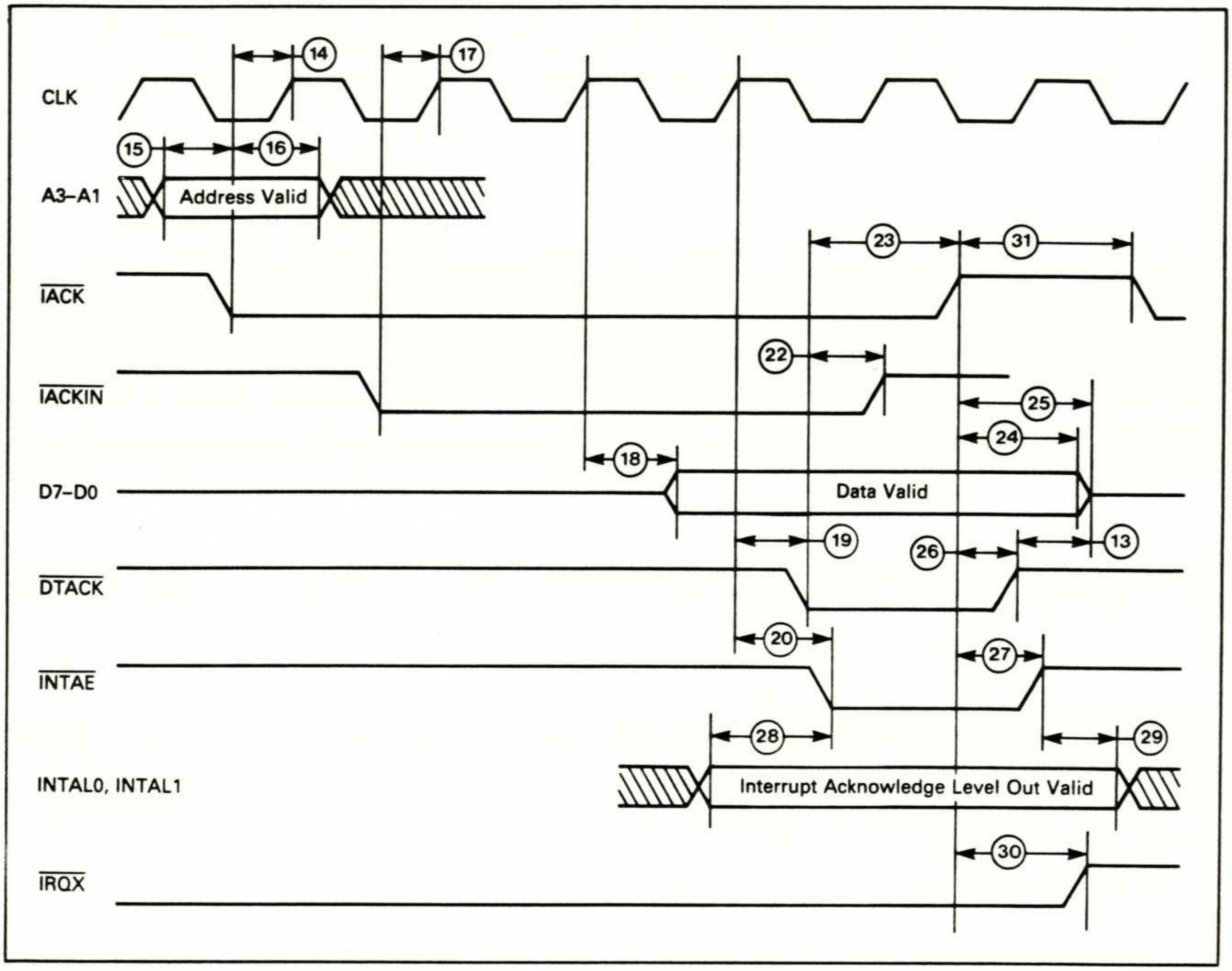
5. $\overline{\text{IACKIN}}$ asserted.

The internal interrupt acknowledge cycle timing is shown in Figure 10. The 8-bit interrupt acknowledge vector is presented to the data bus and $\overline{\text{DTACK}}$ is asserted. Note also that INTAL0 and INTAL1 are valid and $\overline{\text{INTAE}}$ is asserted during this cycle although they would normally not be used. The cycle is terminated (data and $\overline{\text{DTACK}}$ released) after $\overline{\text{IACK}}$ is negated.

During the IACK cycle, the INTERRUPT AUTO-CLEAR control bit (IRAC) comes into play. If the IRAC = one for the responding interrupt source, the INTERRUPT ENABLE (IRE) bit is automatically cleared during the IACK cycle, thus disabling the associated interrupt input and any $\overline{\text{IRQX}}$ output asserted due to this interrupt input. Before another interrupt can be requested from this source, IRE must be set to one by writing to the control register.

Note that $\overline{\text{IACKOUT}}$ is not asserted because this device is responding to the IACK and does not pass the daisy chain signal on. Also, new device interrupt requests occurring on $\overline{\text{INT0}}$–$\overline{\text{INT3}}$ after $\overline{\text{IACK}}$ is asserted are locked out to prevent any race conditions on the daisy chain.

FIGURE 10 — INTERRUPT ACKNOWLEDGE CYCLE — INTERNAL VECTOR



External Interrupt Acknowledge

For an external interrupt acknowledge, the same conditions as listed above are met with one exception. Control register bit X/IN of matching interrupt source must be set to one. The timing is shown in Figure 11. For this cycle, the interrupt vector and DTACK must be supplied by an external device. INTAE is asserted indicating that INTAL0 and INTAL1 are valid. The external device can use these signals to enable the vector and DTACK. The cycle is terminated after IACK is negated.

The IRAC control bit acts in the external interrupt acknowledge the same as described for the internal response (see above). Also, IACKOUT is not asserted and new device interrupts are disabled for reasons discussed above.

Pass On IACK Daisy Chain

If the MC68153 has no interrupt request pending at the same level as the interrupt acknowledge, the IACK daisy chain signal is passed on to the next device if IACKIN is asserted. The following conditions are thus met:

1. IACK asserted.
2. No match exists between [A3, A2, A1] and the [L2, L1, L0] field of an enabled, requesting control register.
3. IACKIN is asserted.

IACKOUT is asserted if these conditions are valid. This output drives IACKIN of the next Interrupter on the daisy chain, passing the signal along. Figure 12 shows the timing for this case. IACKOUT is negated after IACK is negated.

FIGURE 11 — INTERRUPT ACKNOWLEDGE CYCLE — EXTERNAL VECTOR

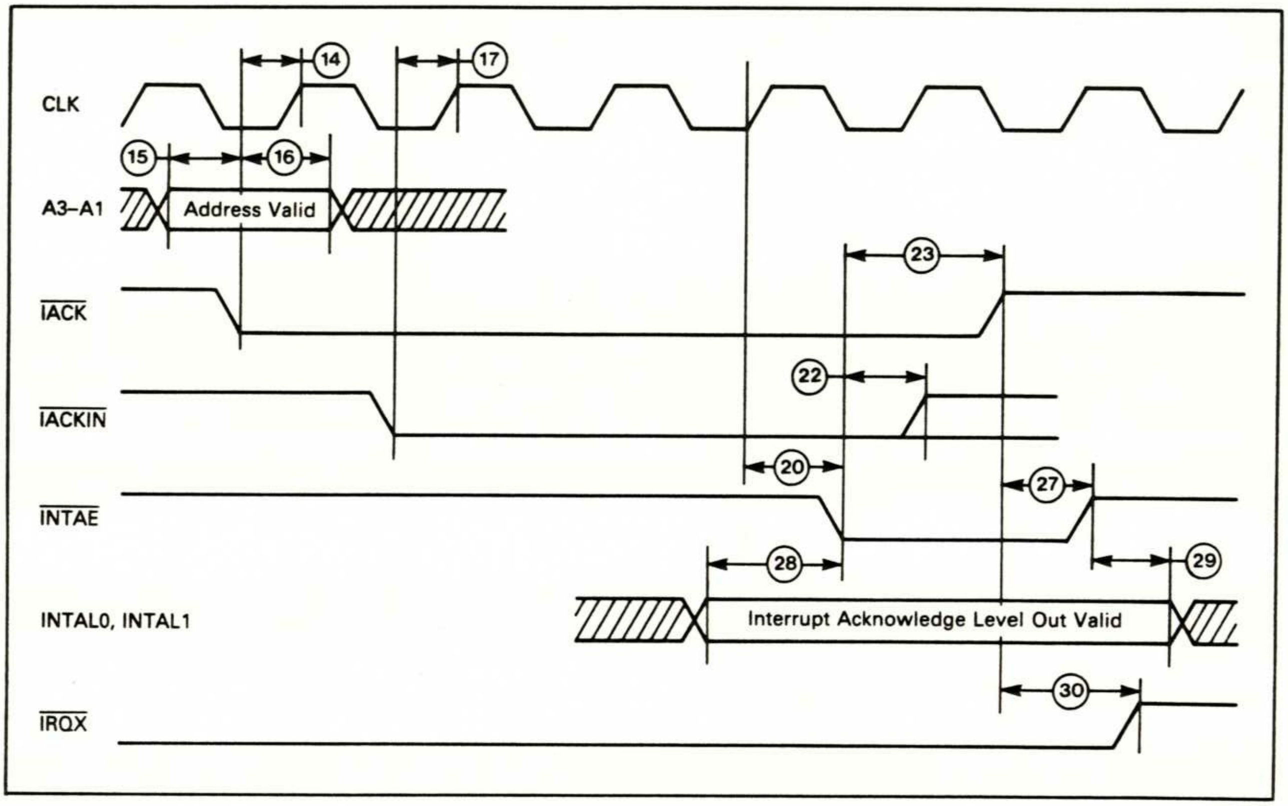
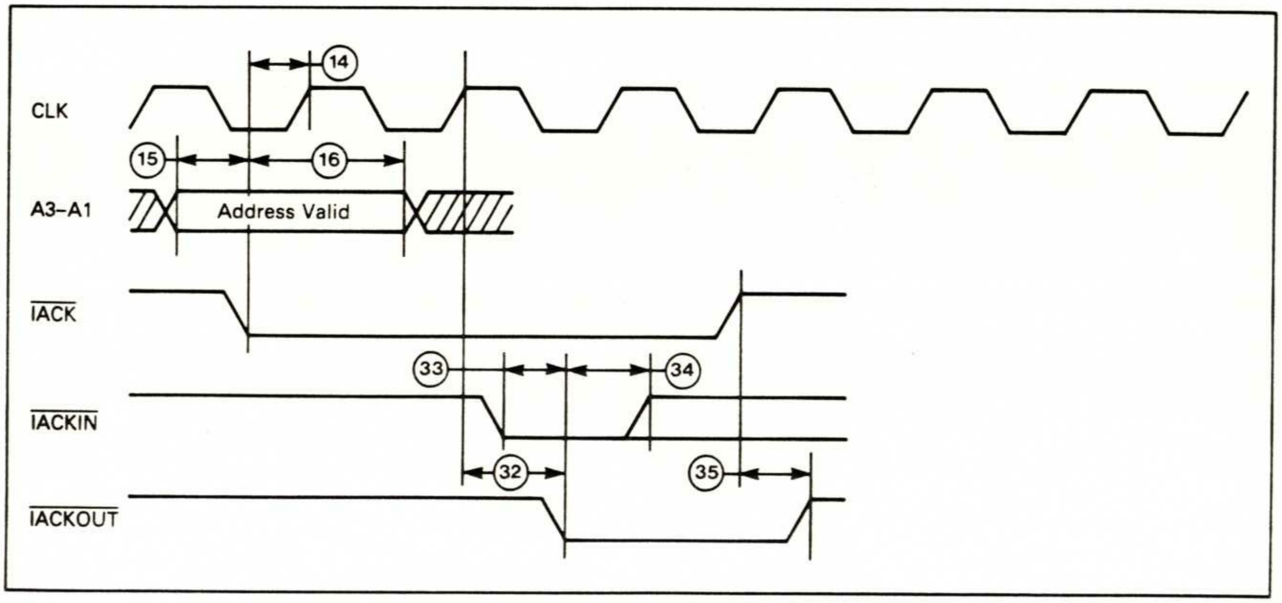


FIGURE 12 — INTERRUPT ACKNOWLEDGE CYCLE — IACKOUT



CONTROL REGISTER FLAGS

Each control register contains a Flag bit (F) and a Flag Auto-Clear bit (FAC). Both bits can be read or altered via a register write without affecting the interrupt operation of the device. The Flag is useful as a status indicator for resource management and as a semaphor in multitasking or multiprocessor systems. Flag (F) is located in bit position 7 and can be used with the MC68000 Test and Set (TAS) instruction.

The Flag Auto-Clear (FAC) is used to manipulate the Flag bit. If the Flag is set to one and the FAC is also one, an interrupt acknowledge cycle to the associated interrupt source clears the Flag bit. This feature is useful in determining the interrupt status and passing messages.

RESET

There is no reset input, however, a chip reset is activated by asserting both $\overline{\text{CS}}$ and $\overline{\text{IACK}}$ simultaneously (Figure 13). These inputs should be held low for a minimum of two clock cycles for a full reset function. The control registers are reset to all zeroes and the Vector Registers are set to a value of $0F. This vector value is the uninitialized vector for the MC68000. See the MC68000 Users Manual for more details on this vector.

CLOCK

The chip clock is required for internal operation to occur. Typical frequency is 16 MHz in VMEbus and VERSAbus applications derived from the system clock. Any frequency can be used, however, up to 25 MHz (Figure 14).

FIGURE 13 — RESET

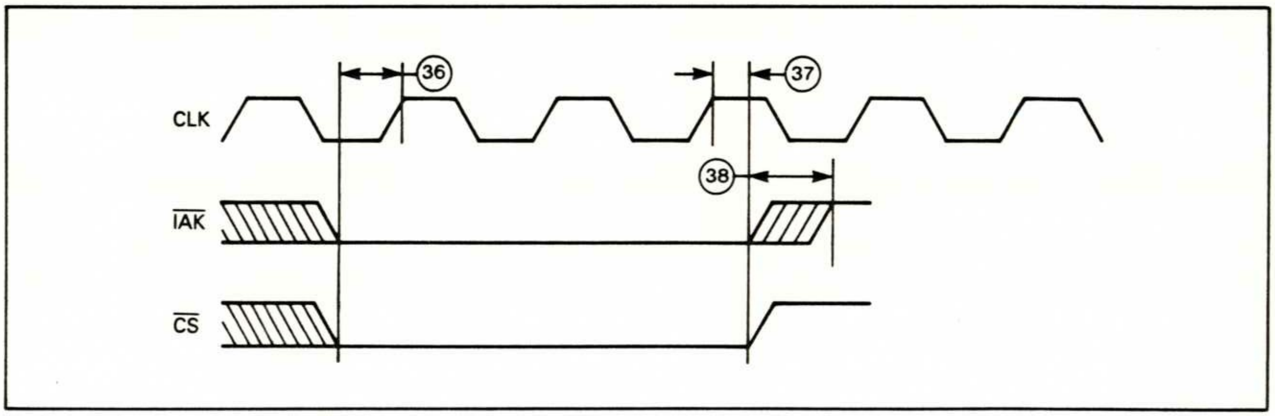


FIGURE 14 — CLOCK WAVEFORM

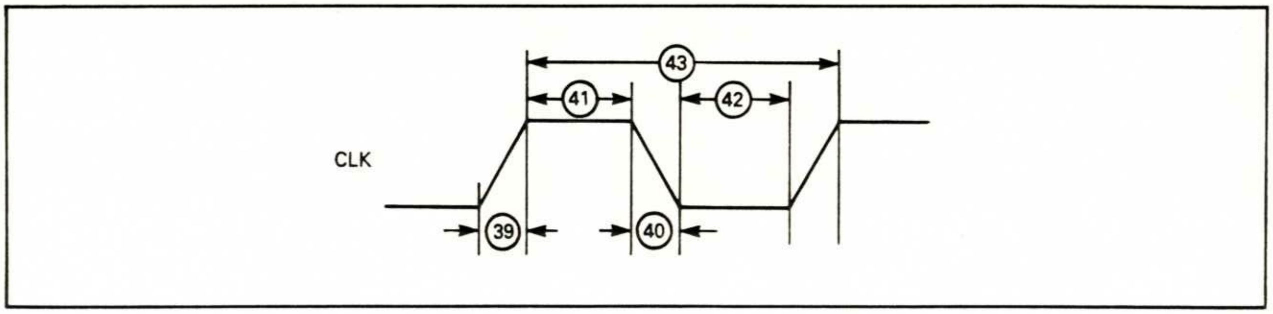


TABLE 1
AC PERFORMANCE SPECIFICATIONS
(V_{CC} = 5.0 V ±5%, T_A = 0°C to 70°C)

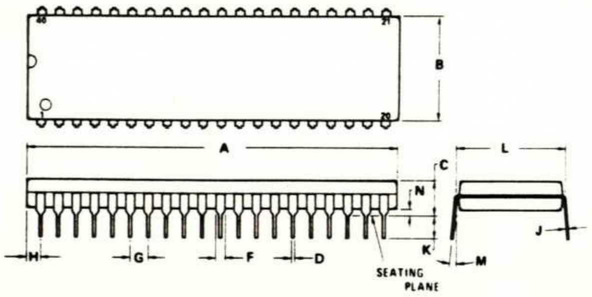
Number	Characteristic	Min	Max	Units	Notes
1	R/$\overline{\text{W}}$, A1–A3 Valid to $\overline{\text{CS}}$ Low (Setup Time)	10	—	ns	
2	$\overline{\text{CS}}$ Low to R/$\overline{\text{W}}$, A1–A3 Invalid (Hold Time)	5.0	—	ns	
3	$\overline{\text{CS}}$ Low to CLK High (Setup Time)	15	—	ns	1
4	CLK High to Data Out Valid (Delay)	—	55	ns	2
5	CLK High to $\overline{\text{DTACK}}$ Low (Delay)	—	40	ns	2
6	$\overline{\text{DTACK}}$ Low to $\overline{\text{CS}}$ High	0		ns	
7	$\overline{\text{CS}}$ High to $\overline{\text{DTACK}}$ High (Delay)	—	35	ns	10
8	$\overline{\text{CS}}$ High to Data Out Invalid (Hold Time)	0	—	ns	
9	$\overline{\text{CS}}$ High to Data Out High-Impedance (Hold Time)	—	50	ns	
10	$\overline{\text{CS}}$ High to $\overline{\text{CS}}$ or $\overline{\text{IACK}}$ Low	20	—	ns	
11	Data In Valid to $\overline{\text{CS}}$ Low (Setup Time)	10	—	ns	
12	$\overline{\text{CS}}$ Low to Data In Invalid (Hold Time)	5.0	—	ns	
13	$\overline{\text{DTACK}}$ High to Data Out High-Impedance	—	25	ns	10
14	$\overline{\text{IACK}}$ Low to CLK High (Setup Time)	15	—	ns	1
15	A1–A3 Valid to $\overline{\text{IACK}}$ Low (Setup Time)	10	—	ns	
16	$\overline{\text{IACK}}$ Low to A1–A3 Invalid (Hold Time)	5.0	—	ns	
17	$\overline{\text{IACKIN}}$ Low to CLK High (Setup Time)	15	—	ns	1, 8
18	CLK High to Data Out Valid (Delay)	—	55	ns	3
19	CLK High to $\overline{\text{DTACK}}$ Low (Delay)	—	40	ns	3
20	CLK High to $\overline{\text{INTAE}}$ Low (Delay)	—	40	ns	3
22	$\overline{\text{DTACK}}$ Low to $\overline{\text{IACKIN}}$ High	0	—	ns	8
23	$\overline{\text{DTACK}}$ Low to $\overline{\text{IACK}}$ High	0	—	ns	
24	$\overline{\text{IACK}}$ High to Data Out Invalid (Hold Time)	0	—	ns	
25	$\overline{\text{IACK}}$ High to Data Out High Impedance (Delay)	—	60	ns	
26	$\overline{\text{IACK}}$ High to $\overline{\text{DTACK}}$ High (Delay)	—	45	ns	10
27	$\overline{\text{IACK}}$ High to $\overline{\text{INTAE}}$ High (Delay)	—	35	ns	
28	INTAL0, INTAL1 Valid to $\overline{\text{INTAE}}$ Low (Setup Time)	1.0	2.0	CLK Per	
29	$\overline{\text{INTAE}}$ High to INTAL0, INTAL1 Invalid (Hold Time)	1.0	2.0	CLK Per	
30	$\overline{\text{IACK}}$ High to $\overline{\text{IRQx}}$ High (Delay)	—	50	ns	7, 10
31	$\overline{\text{IACK}}$ High to $\overline{\text{IACK}}$ or $\overline{\text{CS}}$ Low	20	—	ns	
32	CLK High to $\overline{\text{IACKOUT}}$ Low (Delay)	—	40	ns	5
33	$\overline{\text{IACKIN}}$ Low to $\overline{\text{IACKOUT}}$ Low (Delay)	—	30	ns	4, 8
34	$\overline{\text{IACKOUT}}$ Low to $\overline{\text{IACKIN}}$, $\overline{\text{IACK}}$ High	0	—	ns	8
35	$\overline{\text{IACK}}$ High to $\overline{\text{IACKOUT}}$ High (Delay)	—	35	ns	
36	$\overline{\text{IACK}}$ and $\overline{\text{CS}}$ both Low to CLK High (Setup Time)	15	—	ns	9
37	CLK High to $\overline{\text{IACK}}$ or $\overline{\text{CS}}$ High (Hold Time)	0	—	ns	
38	$\overline{\text{IACK}}$ or $\overline{\text{CS}}$ High to $\overline{\text{IACK}}$ and $\overline{\text{CS}}$ High (Skew)	—	1.0	CLK Per	6
39	Clock Rise Time	—	10	ns	
40	Clock Fall Time	—	10	ns	
41	Clock High Time	20	—	ns	
42	Clock Low Time	20	—	ns	
43	Clock Period	40	—	ns	

NOTES:

1. This specification only applies if the VBIM had completed all operations initiated by the previous bus cycle when $\overline{\text{CS}}$ or $\overline{\text{IACK}}$ was asserted. Following a normal bus cycle, all operations are completed within 2 clock cycles after $\overline{\text{CS}}$ or $\overline{\text{IACK}}$ have been negated. If $\overline{\text{IACK}}$ or $\overline{\text{CS}}$ is asserted prior to completion of these operations, the new cycle, and hence, $\overline{\text{DTACK}}$ is postponed.

   If the $\overline{\text{IACK}}$, $\overline{\text{IACKIN}}$ or $\overline{\text{CS}}$ setup time is violated, $\overline{\text{DTACK}}$ may be asserted as shown, or may be asserted one clock cycle later (i.e. $\overline{\text{IACK}}$ will not be recognized until the next rising edge of the clock).
2. Assumes that 3 has been met.
3. Assumes that 14 and 17 have both been met.
4. Assumes that 14 has been met. ($\overline{\text{IACKOUT}}$ cannot go low prior to $\overline{\text{IACKIN}}$ going low).
5. Assumes that 14 has been met and $\overline{\text{IACKIN}}$ has been low for at least the amount of time specified by 33.
6. 38 is the minimum skew between the last moment when both $\overline{\text{IACK}}$ and $\overline{\text{CS}}$ are asserted to when both are negated, to insure that an access cycle is not unintentionally started.
7. Assumes no other $\overline{\text{INTx}}$ input is causing $\overline{\text{IRQx}}$ to be driven low.
8. In non-daisy chain systems, $\overline{\text{IACKIN}}$ may be tied low.
9. Failure to meet this spec. causes RESET to be ignored for 1 clock period. It is then necessary to keep these signals low for 3 clock periods instead of 2.
10. Delay time is specified from Input signal to Open-Collector Output pulled High thru 1.0 kΩ resistor to +6.5 V.

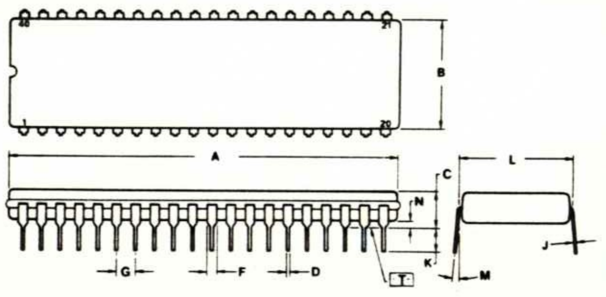
OUTLINE DIMENSIONS



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	51.69	52.45	2.035	2.065
B	13.72	14.22	0.540	0.560
C	3.94	5.08	0.155	0.200
D	0.36	0.56	0.014	0.022
F	1.02	1.52	0.040	0.060
G	2.54 BSC		0.100 BSC	
H	1.65	2.16	0.065	0.085
J	0.20	0.38	0.008	0.015
K	2.92	3.43	0.115	0.135
L	15.24 BSC		0.600 BSC	
M	0°	15°	0°	15°
N	0.51	1.02	0.020	0.040

NOTES:
1. POSITIONAL TOLERANCE OF LEADS (D), SHALL BE WITHIN 0.25 mm (0.010) AT MAXIMUM MATERIAL CONDITION, IN RELATION TO SEATING PLANE AND EACH OTHER.
2. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSION B DOES NOT INCLUDE MOLD FLASH.

CASE 711-03
PLASTIC PACKAGE



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	51.31	53.24	2.020	2.096
B	12.70	15.49	0.500	0.610
C	4.06	5.84	0.160	0.230
D	0.38	0.56	0.015	0.022
F	1.27	1.65	0.050	0.065
G	2.54 BSC		0.100 BSC	
J	0.20	0.30	0.008	0.012
K	3.18	4.06	0.125	0.160
L	15.24 BSC		0.600 BSC	
M	5°	15°	5°	15°
N	0.51	1.27	0.020	0.050

NOTES:
1. DIM -A- IS DATUM
2. POSITIONAL TOLERANCE FOR LEADS:
   | ⌖ | ø 0.25(0.010) Ⓜ | T | A Ⓜ |
3. -T- IS SEATING PLANE
4. DIM L TO CENTER OF LEADS WHEN FORMED PARALLEL
5. DIMENSIONS A AND B INCLUDE MENISCUS
6. DIMENSIONING AND TOLERANCING PER ANSI Y14.5, 1973

CASE 734-04
CERAMIC PACKAGE

TYPICAL THERMAL CHARACTERISTICS

Package	θJA (Junction to Ambient) Still Air	Junction Temperature Still Air @ 70°C Ambient
L Suffix	40°C/W	147°C
P Suffix[1]	35°C/W	137°C

NOTES:
1. For reliable system operation the maximum allowable junction temperature (T_J) for plastic encapsulated packages has been limited to +140°C. Exceeding this limit will accellerate "wear-out" mechanisms associated with industry standard assembly methods using thermosonic ball bonds to attach gold (Aμ) bond wire to aluminum (Al) bond pads on the die surface.
2. At T_J = 140°C, time to 0.1% failure due to Aμ/Al interconnect = 8,920 Hours.

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein, neither does it convey any license under its patent rights nor the rights of others. Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Employment Opportunity/Affirmative Action Employer.

MOTOROLA *Semiconductor Products Inc.*

BOX 20912 • PHOENIX, ARIZONA 85036 • A SUBSIDIARY OF MOTOROLA INC.

16542-3 PRINTED IN USA 9-84 IMPERIAL LITHO C24687 12,000

ADI-1057

Signetics

SCB68430
Direct Memory Access Interface (DMAI)

Preliminary Specification

Microprocessor Products

DESCRIPTION

The SCB68430 Direct Memory Access Interface (DMAI) is a single channel interface circuit which is intended to complement the performance and architectural capabilities of the SCN68000 microprocessor. The DMAI functions by transferring a series of operands (data) between memory and a device: operand sizes may be byte, word, or long word. A block is a sequence of operands: the number of operands in the block is determined by a transfer count stored within the DMAI. The SCB68430 can be programmed to utilize single cycle (cycle stealing) or burst data transfers.

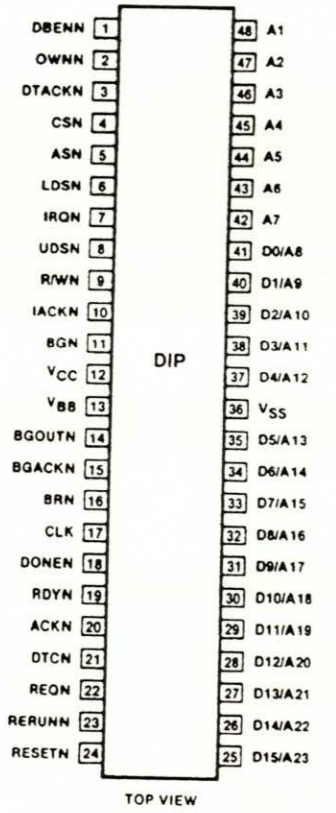
The DMAI provides two interfaces. The microprocessor interface is fully compatible with the SCN68000 microprocessor. The device interface includes lines for requesting, acknowledging, controlling, and timing the data transfers. The DMAI is a single-channel subset of the other 68000 family DMA controllers (68440 and 68450). It is software compatible with these devices and provides similar interfacing signals to both the system bus and the device.

The SCB68430 is constructed using Signetics ISL bipolar technology.

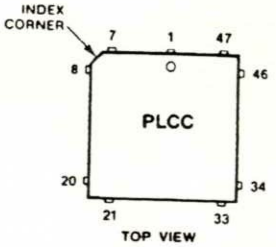
FEATURES

- **Bus compatible with SCN68000 microprocessor**
- **Software compatible with other 68K family DMA controllers**
- **Single address transfers**
- **Cycle steal and burst mode operation**
- **Bus arbitration daisy chain**
- **Automatic rerun on bus error**
- **Supports 32-bit transfers for VME bus**
- **Supports SCN68000 vectored interrupts**
- **24-bit address counter**
- **16-bit transfer counter**
- **Maximum transfer rate of 5 Mbytes per second**
- **Signetics ISL bipolar technology**

PIN CONFIGURATION



CD00650S



CD0052PS

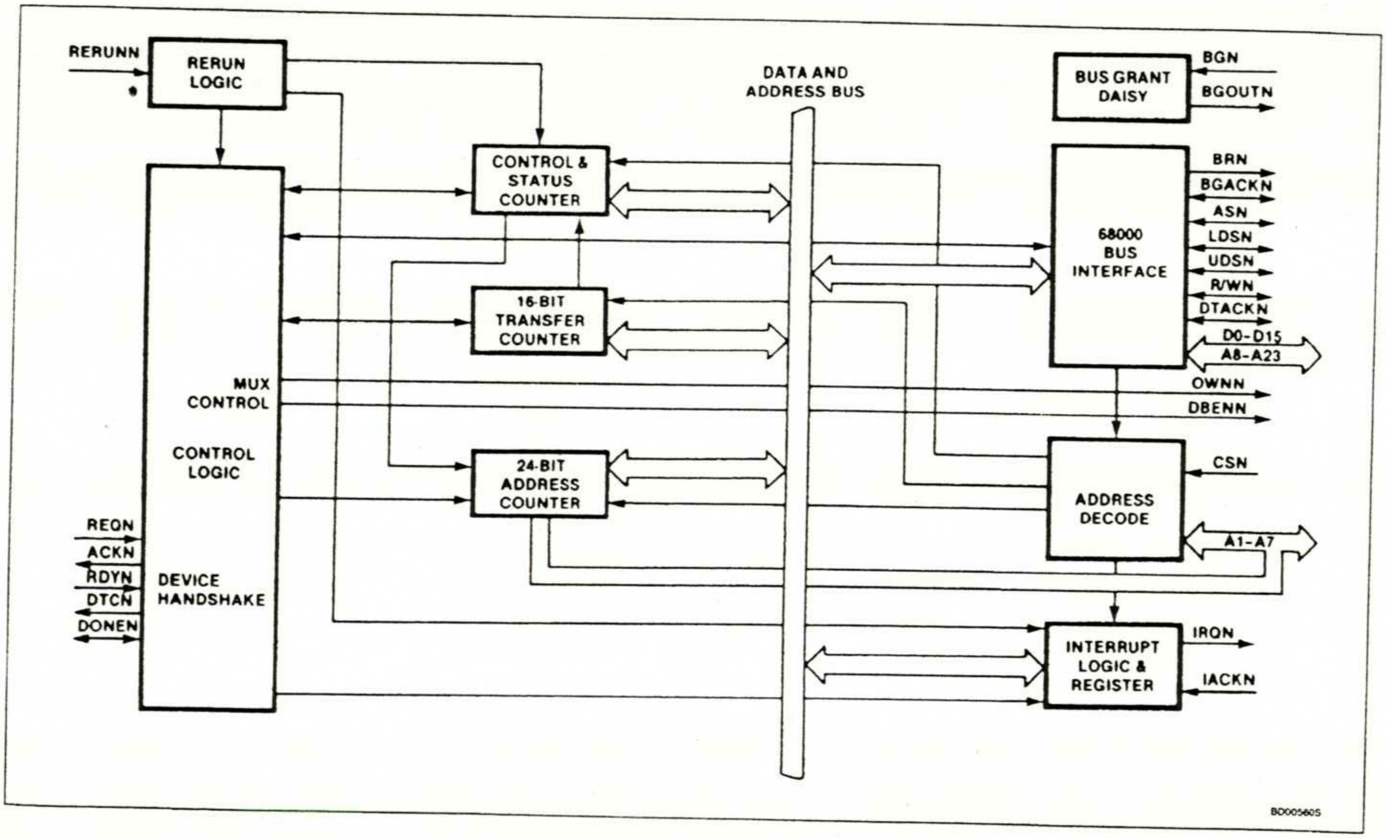
Pin	Function	Pin	Function	Pin	Function
1	NC	19	CLK	36	D7/A15
2	DBENN	20	DONEN	37	D6/A14
3	OWNN	21	RDYN	38	D5/A13
4	DTACKN	22	ACKN	39	V_{SS}
5	CSN	23	DTCN	40	NC
6	ASN	24	REQN	41	D4/A12
7	LDSN	25	RERUNN	42	D3/A11
8	IRQN	26	RESETN	43	D2/A10
9	UDSN	27	NC	44	D1/A9
10	R/WN	28	D15/A23	45	D0/A8
11	IACKN	29	D14/A22	46	A7
12	BGN	30	D13/A21	47	A6
13	V_{CC}	31	D12/A20	48	A5
14	NC	32	D11/A19	49	A4
15	V_{BB}	33	D10/A18	50	A3
16	BGOUTN	34	D9/A17	51	A2
17	BGACKN	35	D8/A16	52	A1
18	BRN				

Direct Memory Access Interface (DMAI) SCB68430

ORDERING CODE

PACKAGES	$V_{CC} = 5V \pm 5\%$, $T_A = 0°C$ to 70°C	
	10MHz	12.5MHz
Ceramic DIP	SCB68430CAI48	SCB68430CCI48
Plastic DIP	SCB68430CAN48	SCB68430CCN48
Plastic LCC	SCB68430CAA52	SCB68430CCA52

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	DESCRIPTION
A1 – A7	48 – 42	I/O	**Address Lines:** Active high, three-state. In the MPU mode, these low order address lines specify which internal register of the DMAI is being accessed. In DMA mode, A1 – A7 are ouputs which provide the low order address bits of the location being accessed. Three-stated in IDLE Mode.
A8 – A23/ D0 – D15	41 – 37 35 – 25	I/O	**Address/Data Lines:** Active high, three-state. These lines are time multiplexed for data and address leads. The lines OWNN, RWN, CSN, and DBENN are used to control the demultiplexing of the address and data using external circuitry. In MPU mode, the bidirectional data lines (D0 – D15) are used to transfer data between the MPU and the DMAI. In the DMA mode, A8 – A23 provide the high order address bits of the location being accessed. Three-stated in IDLE mode.
ASN	5	I/O	**Address Strobe:** Active low, three-state. In MPU and IDLE modes, ASN is an input which indicates that the current bus master has placed a valid address on the bus. It is monitored by the DMAI during bus arbitration to ascertain that the previous bus master has completed the current bus cycle. In DMA mode, it is an output indicating that the DMAI has placed a valid address on the bus.
UDSN	8	I/O	**Upper Data Stobe:** Active low, three-state. In MPU and IDLE modes, UDSN is an input which indicates that the upper data byte of the addressed word is being addressed. In DMA mode, it is an output with the same meaning.
LDSN	6	I/O	**Lower Data Strobe:** Active low, three-state. In MPU and IDLE modes, LDSN is an input which indicates that the lower data byte of the addressed word is being addressed. In DMA mode, it is an output with the same meaning.
R/WN	9	I/O	**Read/Write:** Active high for read, low for write, three-state. In MPU mode, R/WN is an input which controls the direction of data flow through the DMAI's input/output data bus interface and, if required, through an external data bus buffer. R/WN high causes the DMAI to place the data from the addressed register on the data bus, while R/WN low causes the DMAI to accept data from the data bus. In DMA mode, R/WN is an output to memory and I/O controllers indicating the type of bus cycle. It is held three-stated during IDLE mode.
CSN	4	I	**Chip Select:** Active low. When low, places the DMAI into the MPU mode. This input signal is used to select the DMAI for programmed data transfers. These transfers take place over D0 – D15 as controlled by the R/WN and A1 – A7 inputs. The DMAI is deselected when CSN is high. CSN is ignored during DMA mode.
DTACKN	3	I/O	**Data Transfer Acknowledge:** Active low, three-state. In MPU mode, DTACKN is asserted on a write cycle to indicate that the data on the bus has been latched, and on a read cycle or interrupt acknowledge cycle to indicate that valid data is present on the bus. The signal is negated (driven high) when completion of the cycle is indicated by negation of the CSN or IACKN input, and returns to the inactive third state a short time after it is negated. In DMA Mode, DTACKN is an input monitored by the DMAI to determine when the addressed device (memory) has latched the data (write cycle) or put valid data on the bus (read cycle).
RESETN	24	I	**Master Reset:** Active low. Assertion of this pin clears internal control registers (See table 1), initializes the interrupt vector register to H'0F', and sets the status register to the default value B'0000 000X', where X is the state of RDYN. All bidirectional I/O lines are three-stated and the DMAI is placed in the IDLE mode.
CLK	17	I	**Clock:** Active high. Usually the system clock, but may be any clock meeting the electrical specifications. Used by the DMAI to synchronize device functions and external control lines, and may not be gated off at any time.
IRQN	7	O	**Interrupt Request:** Active low, open collector. This output is asserted, if interrupts are enabled, upon end of transfer, on occurrence of a bus error, and on receipt of an abort from the MPU. The CPU can read the status register to determine the interrupting condition(s), or can respond with an interrupt acknowledge cycle to cause the DMAI to output an interrupt vector on the data bus.
IACKN	10	I	**Interrupt Acknowledge:** Active low. When asserted, indicates that the current cycle is an interrupt acknowledge cycle. The DMAI normally responds by placing the contents of the interrupt vector register on the data bus and asserting DTACKN. IACKN is not serviced if the DMAI has not generated an interrupt request.
BRN	16	O	**Bus Request:** Active low, open collector. BRN is asserted by the DMAI to request ownership of the bus after a DMA request is sensed on the REQN input from the I/O device. It is negated when the bus has been granted (BGN low) and BGACKN has been asserted, or, in burst DMA request mode, if the I/O device negates its request at least one clock cycle before BGACKN is asserted.
BGN	11	I	**Bus Grant:** Active low. BGN indicates to the DMAI that it is to be the next bus master. This signal is originated by the MPU and propagated via a daisy chain or other prioritization mechanism. After BGN is asserted, the DMAI waits until DTACKN, ASN, and BGACKN have become inactive before assuming ownership of the bus by asserting BGACKN.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.	TYPE	DESCRIPTION
BGOUTN	14	O	**Bus Grant Output:** Active low. Daisy chain output which is asserted by the DMAI when BGN is asserted and the DMAI does not have a bus request pending.
BGACKN	15	I/O	**Bus Grant Acknowledge:** Active low, three-state. As an input, BGACKN is monitored by the DMAI during the bus arbitration cycle to determine when it can assume ownership of the bus (BGACKN negated). In DMA mode, it is asserted by the DMAI to indicate that it is the bus master. Three-stated in MPU and IDLE modes.
RERUNN	23	I	**Rerun:** Active low. This input is asserted by external error detect logic to indicate a bus error. In DMA mode, the DMAI stops operation and three-states the data, address, and control lines, except BGACKN. It remains halted until RERUNN becomes inactive, and then re-tries the last bus cycle. If RERUNN is asserted again, the DMAI sets the ERR bit in the status register, stops DMA operation, releases the bus, and interrupts the CPU, if interrupts are enabled, responding with a special interrupt vector when IACKN is asserted. Not monitored in MPU and IDLE modes.
REQN	22	I	**DMA Request:** Active low. This input from the I/O device requests service from the DMAI and causes the DMAI to request control of the bus. In burst mode, the input is level sensitive, and the DMAI releases the bus after REQN is negated and the current DMA cycle is completed. In cycle steal mode, the REQN input is negative edge triggered. A negative going edge must occur at least one clock cycle before DTCN is asserted to accomplish continuous transfer cycles but not earlier than beginning of master cycle.
ACKN	20	O	**DMA Request Acknowledge:** Active low. ACKN is asserted by the DMAI to indicate that it has gained the bus and the requested bus cycle is now beginning. It is asserted at the beginning of every bus cycle after ASN has been asserted, and is negated at the end of every bus cycle.
RDYN	19	I	**Device Ready:** Active low. RDYN is asserted by the requesting device to indicate to the DMAI that valid data has either been stored or put on the bus. If negated, it indicates that the data has not been stored or presented, causing the DMAI to enter wait states. RDYN can be held low continuously if the device is fast enough so that wait states are not required.
DTCN	21	O	**Device Transfer Complete:** Active low. In DMA mode, DTCN is asserted by the DMAI to indicate to the device that the requested data transfer is complete. On a write to memory operation, it indicates that the data provided by the device has been successfully stored. On a read from memory operation, it indicates to the device that the data from memory is present on the data bus and should be latched.
DONEN	18	I/O	**Done:** Active low, open collector. As an output, DONEN is asserted by the DMAI concurrent with the ACKN output to indicate to the device that the transfer count is exhausted and that the DMAI's operation is completed as a result of that transfer. As an input, if asserted by the device before the transfer count became zero, it causes the DMAI to abort service and generate an interrupt request, if interrupts are enabled.
OWNN	2	O	**Own:** Active low, open collector. This output is asserted by DMAI during the DMA mode to indicate bus mastership. It can be used to enable external address/data and control buffers. Inactive in MPU and IDLE modes.
DBENN	1	O	**Data Bus Enable:** Active low, open collector. Asserted by the DMAI when CSN is asserted or when IACKN is asserted and the DMAI has an interrupt request pending. Can be used to enable bidirectional data buffers for D0 – D15. Inactive in IDLE and DMA mode.
V_{CC}	12	I	**Power Supply:** +5 volt power input.
V_{BB}	13	I	**Power Supply:** +1.5 volt power input.
V_{SS}	36	I	**Ground:** Signal and power ground input.

PIN DESCRIPTION

The Pin Description table describes the function of each of the pins of the DMAI. Signal names ending in 'N' are active low. All other signals are active high. In the descriptions, 'MPU mode' refers to the state when the DMAI is chip selected. The term 'DMA mode' refers to the state when the DMAI assumes ownership of the bus. The DMAI is in the 'IDLE mode' at all other times.

In this data sheet signals are discussed using the terms 'active' and 'inactive' or 'asserted' and 'negated' independent of whether the signal is active in the high (logic one) state or the low (logic zero) state. Refer to the individual pin descriptions for the definition of the active level of each signal.

REGISTERS AND COUNTERS

Register Map

The internal accessible register organization of the DMAI is shown in table 1. The following rules apply to all registers:

1. A read from a reserved location in the map results in a read from the 'null register'. The null register returns all ones for data and results in a normal bus cycle. A write to one of these locations results in a normal bus cycle but no write occurs.
2. Unused bits of a defined register are read as indicated in the register descriptions.
3. All registers are addressable as 8-bit quantities. To facilitate operation with the 68K MOVEP instruction, addresses are ordered such that certain sets of registers may also be accessed as words or long words.

Direct Memory Access Interface (DMAI) · SCB68430

The operation of the DMAI is programmed by writing control words into the appropriate registers. Operational feedback is provided via status registers which can be read by the CPU. The contents of certain control and status registers are initialized on RESET.

To provide compatibility with the other 68K family DMA controllers, control and status bits are mapped in bit positions equivalent to where they are located in the register map of the other devices. Bits which are used in the other devices but not in the DMAI are assigned default values. If upward compatibility to the other controllers is required, the programmer should use these default values when writing the control words to the registers, although they have no effect in the DMAI. When a register is read, the default value is returned regardless of the value used when the register is programmed. The default value is indicated by '(x)' in unused bit positions in the register formats, which are illustrated in table 2.

Table 1. DMAI ADDRESS MAP

ADDRESS BITS[1,2] 7 6 5 4 3 2 1 0	ACRONYM	REGISTER NAME	MODE	AFFECTED BY RESET
d d 0 0 0 0 0 0	CSR	Channel Status Register	R/W[3]	Yes
d d 0 0 0 0 0 1	CER	Channel Error Register	R	Yes
d d 0 0 0 0 1 0		Reserved		
d d 0 0 0 0 1 1		Reserved		
d d 0 0 0 1 0 0	DCR	Device Control Register	R/W	Yes
d d 0 0 0 1 0 1	OCR	Operation Control Register	R/W	Yes
d d 0 0 0 1 1 0	SCR	Sequence Control Register	R/W[4]	No
d d 0 0 0 1 1 1	CCR	Channel Control Register	R/W	Yes
d d 0 0 1 0 0 0		Reserved		
d d 0 0 1 0 0 1		Reserved		
d d 0 0 1 0 1 0	MTCH	Memory Transfer Counter High	R/W	No
d d 0 0 1 0 1 1	MTCL	Memory Transfer Counter Low	R/W	No
d d 0 0 1 1 0 0	MACH	Memory Address Counter High	R/W[4]	No
d d 0 0 1 1 0 1	MACMH	Memory Address Counter Middle High	R/W	No
d d 0 0 1 1 1 0	MACML	Memory Address Counter Middle Low	R/W	No
d d 0 0 1 1 1 1	MACL	Memory Address Counter Low	R/W	No
d d 0 1 d d d d		Reserved		
d d 1 0 0 0 d d		Reserved		
d d 1 0 0 1 0 0		Reserved		
d d 1 0 0 1 0 1	IVR	Interrupt Vector Register – Normal	R/W	Yes
d d 1 0 0 1 1 0		Reserved		
d d 1 0 0 1 1 1	IVR	Interrupt Vector Register – Error	R/W	Yes
d d 1 0 1 0 d d		Reserved		
d d 1 0 1 1 0 0		Reserved		
d d 1 0 1 1 0 1	CPR	Channel Priority Register	R/W[4]	No
d d 1 0 1 1 1 0		Reserved		
d d 1 0 1 1 1 1		Reserved		
d d 1 1 d d d d		Reserved		

NOTES:
1. A0 = 0 for UDSN asserted, A0 = 1 for LDSN asserted.
2. 'd' designates don't care.
3. A write to this register may perform a status resetting operation.
4. This register is a dummy register present only to provide compatibility with other 68K family DMA controllers. A write to this register has no effect on the DMAI.

Table 2. REGISTER BIT FORMATS

DEVICE CONTROL REGISTER

	BIT15	BIT14	BIT13	BIT12	BIT11	BIT10	BIT09	BIT08
DCR	EXTERNAL REQUEST MODE 0 = BURST 1 = CYCLE STEAL	NOT USED (0)	NOT USED (1)	NOT USED (1)	NOT USED (*)	NOT USED (0)	NOT USED (0)	NOT USED (0)

*Should be programmed as '0' for SIZE (OCR[5:4]) = 00 and as '1' otherwise. When read, the value of this bit is OCR[5]. .OR.OCR[4].

OPERATION CONTROL REGISTER (OCR)

	BIT07	BIT06	BIT05	BIT04	BIT03	BIT02	BIT01	BIT00
	DIRECTION		OPERAND SIZE					
OCR	0 = MEM TO DEV 1 = DEV TO MEM	NOT USED (0)	00 = BYTE 01 = WORD (16 BIT) 10 = LONG WORD* 11 = WORD (32-BIT)*		NOT USED (0)	NOT USED (0)	NOT USED (1)	NOT USED (0)

*Long word and 32-bit word modes are not supported by 68440. 32-bit word mode is not supported by 68450.

SEQUENCE CONTROL REGISTER (SCR)

	BIT15	BIT14	BIT13	BIT12	BIT11	BIT10	BIT09	BIT08
SCR	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (1)	NOT USED (0)	NOT USED (0)

CHANNEL CONTROL REGISTER (CCR)

	BIT07	BIT06	BIT05	BIT04	BIT03	BIT02	BIT01	BIT00
	START			SOFTWARE ABORT	INTERRUPT ENABLE			
CCR	0 = NO 1 = YES	NOT USED (0)	NOT USED (0)	0 = NO 1 = YES	0 = NO 1 = YES	NOT USED (0)	NOT USED (0)	NOT USED (0)

CHANNEL STATUS REGISTER (CSR)

	BIT15	BIT14	BIT13	BIT12	BIT11	BIT10	BIT09	BIT08
	CHANNEL OPERATION COMPLETE		NORMAL DEVICE TERMINATE	ERROR	CHANNEL ACTIVE			READY INPUT STATE
CSR	0 = NO 1 = YES	NOT USED (0)	0 = NO 1 = YES	0 = NO 1 = YES	0 = NO 1 = YES	NOT USED (0)	NOT USED (0)	0 = LOW 1 = HIGH

CHANNEL ERROR REGISTER (CER)

	BIT07	BIT06	BIT05	BIT04	BIT03	BIT02	BIT01	BIT00
				ERROR CODE				
CER	NOT USED (0)	NOT USED (0)	NOT USED (0)	00000 = NO ERROR 01001 = BUS ERROR 10001 = SOFTWARE ABORT				

CHANNEL PRIORITY REGISTER (CPR)

	BIT07	BIT06	BIT05	BIT04	BIT03	BIT02	BIT01	BIT00
CPR	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)

Device Control Register (DCR)

[15] External Request Mode

This bit selects whether the DMAI operates in burst or cycle steal mode.

0 Burst mode. This mode allows a device to request the transfer of multiple operands using consecutive bus cycles. In this mode the request (REQN) line is an active low input which is asserted by the device to request an operand transfer. The DMAI services the request by arbitrating for the bus, obtaining the bus, and notifying the peripheral by asserting the acknowledge (ACKN) output. If the request line is active when the DMAI asserts ACKN, and remains active at least until the DMAI asserts device transfer complete (DTCN), the DMAI recognizes a valid request for another operand, which will be transferred during the next bus cycle. If the request line is negated before the DMAI asserts DTCN, the DMAI relinquishes the bus and waits for the next request, but the current transfer will be completed.

1 Cycle steal mode. In this mode, the device requests an operand transfer by generating a falling edge on the request (REQN) line. The DMAI services the request by arbitrating for the bus, obtaining the bus, and notifying the peripheral by asserting the acknowledge (ACKN)

output. The request line must be in the inactive state for at least one clock cycle before a request is made. After a request has been asserted, it must remain at the assertion level for at least one clock cycle. If another request is received before the first operand part of a former request is acknowledged, the second request is not recognized. Normally, the DMAI will relinquish the bus after servicing a valid request. However, if the device generates a new request before the DMAI asserts DTCN for the last operand part, the DMAI will retain ownership of the bus and that request will be serviced before the bus is relinquished.

Operation Control Register (OCR)

[7] Direction

0 Transfer is from memory to device.

1 Transfer is from device to memory.

[5:4] Operand Size

The programming of these bits determine whether UDSN, LDSN, or both are generated during the transfer cycle and the increment by which the memory address counter (MAC) is changed in each transfer cycle.

00 Byte. The operand size is 8 bits. The MAC is incremented by one after each operand transfer. If the LSB of the MAC is a '0', UDSN is asserted during the transfer. If the LSB of a MAC is a '1', LDSN is asserted during the transfer. The transfer counter decrements by one before each byte is transferred.

01 Word. The operand size is 16 bits. The MAC is incremented by two after each operand transfer. The value of the LSB of the MAC is ignored and both UDSN and LDSN are asserted during the transfer. The transfer counter decrements by one before each word is transferred.

10 Long word. The operand size is 32 bits. The operand is transferred as two 16-bit words. The MAC is incremented by two after each 16-bit word is transferred. The value of the LSB of the MAC is ignored and both UDSN and LDSN are asserted during the transfer. The transfer counter decrements by one before the entire long word is transferred. Note that this mode is not implemented in the 68440.

11 Double word. The operand size is 32 bits. The operand is transferred as a single 32-bit word. The MAC is incremented by four after each operand transfer. The value of the two LSBs of the MAC is ignored (the A1 output will always be a zero in this mode) and both UDSN and LDSN are asserted during the transfer. The transfer counter decrements by one before the double word is transferred. Note that this mode is not implemented in the 68440 or 68450; it is included in the DMAI to support VME bus operations.

Sequence Control Register (SCR)

This register serves no function in the DMAI. It is included only to provide compatibility with the programming for the 68440 and 68450 DMA controllers.

Channel Control Register (CCR)

[7] Start Operation

0 No start pending.

1 Start operation. The start bit is set to initiate operation of the DMAI. The memory address counter and the memory transfer counter should have been previously initialized, and all bits of the channel status register (CSR) should have previously been reset. The DMAI initiates operation by clearing any pending requests, clearing the start bit, and setting the channel active bit in the CSR. The DMAI is then ready to receive requests for an operation. The channel cannot be started if any of the internal status bits in the CSR (CSR[15:11]) have not been cleared.

A pending start cannot be reset by a write to the register. START can be cleared only by the DMAI when it starts operation or by setting the software abort bit (CCR[4]).

[4] Software Abort

0 Do not abort.

1 Abort operation. Setting this bit terminates the current operation of the DMAI and places it in the IDLE state. The channel operation complete and error bits in the CSR are set, the channel active bit in the CSR is reset, and an ABORT ERROR condition is signaled in the CER. Setting this bit causes a pending start to be reset.

[3] Interrupt Enable

0 Interrupts not enabled.

1 Enable interrupts. An interrupt request is generated if the channel operation complete bit in the CSR is set. When the IACKN input is asserted, the DMAI returns the normal interrupt vector if the error bit in the CSR is not set, or the error interrupt vector if error is set.

Channel Status Register (CSR)

A read of this register provides the status of the DMAI. The COC, NDT, and ERR bits can be cleared by writing a '1' to the bit positions of the register which are to be cleared. Those bit positions which are written with a '0' remain unaffected.

[15] Channel Operation Complete

This bit is set following the termination, whether successful or not, of any DMAI operation and indicates that the DMA transfer has completed. This bit must be cleared to start another channel operation.

[13] Normal Device Termination

This bit is set when the device terminates the DMAI operation by asserting the DONEN line while the device was being acknowledged. This bit must be cleared to start another channel operation.

[12] Error

This bit is used to report that the DMAI's operation was terminated due to the occurrence of an error. The condition which caused the error can be determined by reading the channel error register (CER). This bit must be cleared to start another channel operation. When this bit is cleared, the CER is also cleared.

[11] Channel Active

This bit is set after the channel has been started and remains set until the channel operation terminates. It is then automatically cleared by the DMAI. The bit is unaffected by the write operations.

[8] Ready Input State

This bit reflects the state of the RDYN input at the time the CSR is read. The bit is a '0' if RDYN is low and a '1' if RDYN is high. This bit is unaffected by write or reset operations.

Channel Error Register (CER)

[4:0] Error Code

This field indicates the source of error when an error is indicated in CER[12]. The contents of this register are cleared when CER[12] is cleared.

00000 No error.

01001 Bus error. A bus error occurred during the last bus cycle generated by the DMAI. See rerun description in OPERATION section.

10001 Software abort. The channel operation was terminated by a software abort command. See CCR[4].

Channel Priority Register (CPR)

This register serves no function in the DMAI. It is included only to provide compatibility with the programming for the other 68K family DMA controllers.

Memory Address Counter (MACH, MACMH, MACML, MACL)

The 32-bit memory address counter is used to program the memory location where the first operand to be transferred is located or is to be transferred to, depending on the direction of transfer. The counter must be initialized prior to beginning the transfer of a block of data and then increments automatically depending on the operand length, as described in the Operation Control Register description.

Only the least significant 24 bits of the counter (MACMH, MACML, and MACL) are implemented in the DMAI. The most significant byte of the counter, MACH, is provided only to allow compatibility with programming of the 68440 and 68450. Writing to MACH has no effect on the DMAI operation. Reading MACH always returns H'00'.

Memory Transfer Counter (MTCH, MTCL)

The 16-bit memory transfer counter programs the number of operands to be transferred by the DMAI. The counter must be initialized prior to beginning the transfer of a block of data and then decrements once per operand transfer (regardless of operand size) until it reaches the terminal value of zero. Channel operation then terminates and the COC bit in the CSR will be asserted.

Interrupt Vector Register (IVR)

The IVR contains the value to be placed on the data bus upon receipt of an interrupt acknowledge from the MPU. Only the seven most significant bits of the programmed value are used by the DMAI. The output vector from the DMAI contains a zero in the least significant bit position if a normal termination occurred (error bit not set) and contains a one in the least significant bit position if termination was due to an error (error bit set).

The contents of this register are initialized to H'0F' by a reset. The value returned will be H'0F', regardless of the error state, until the register is programmed by the MPU.

To provide compatibility with the other 68K family DMA controllers, the IVR has two addresses (see table 1). If program compatibility is required, the value written at the normal IVR address should have a zero as its LSB, and the value written at the error IVR address should be the same but with the LSB equal to one.

OPERATION

A DMAI operation proceeds in three principal phases. During the initialization phase, the MPU configures the channel control registers, loads the initial memory address and transfer count, and starts the channel. During the transfer phase, the DMAI accepts requests for transfers from the device, arbitrates for and acquires ownership of the bus, and provides for addressing and bus controls for the transfers. The termination phase occurs after the operation is complete, when the DMAI reports the status of the operation.

Operation Initiation

After having programmed the control registers, the memory address counter, and the memory transfer counter, the MPU sets the start bit (CCR[7]). The DMAI initiates the operation by clearing any pending requests, clearing the start bit, and setting the channel active bit in the CSR. The DMAI is then ready to receive valid requests for an operation.

The channel cannot be started if any of the internal status bits in the CSR (CSR[15:11]) have not been cleared. An error is not signaled if this condition occurs. The only indication of this state is that the start bit remains set in the CCR. A pending start cannot be reset by a write to the register. START can be cleared only by the DMAI when it starts operation or by setting the software abort bit (CCR[4]).

Device/DMAI Communication

Communication between the peripheral device and the DMAI is accommodated by five signal lines:

Request (REQN)

The device makes a request for service by asserting the request line. The DMAI can operate in either the burst request mode or the cycle stealing request mode, as programmed by the external request mode bit (DCR[15]).

The burst mode allows a device to request the transfer of multiple operands using consecutive bus cycles. In this mode the request line is an active low input. The DMAI services the request by arbitrating for the bus, obtaining the bus, and notifying the peripheral by asserting the acknowledge (ACKN) output. If the request line is active when the DMAI asserts ACKN, and remains active at least until the DMAI asserts device transfer complete (DTCN), the DMAI recognizes a valid request for another operand, which will be transferred during the next bus cycle. If the request line is negated before the DMAI asserts DTCN, the DMAI relinquishes the bus and waits for the next request. For long word transfers (2 x 16), the request must be asserted at least until the acknowledge for the second part of the operand has been asserted.

In the cycle steal mode, the device requests an operand transfer by generating a falling edge on the request line. The DMAI services the request by arbitrating for the bus, obtaining the bus, and notifying the peripheral by asserting the acknowledge (ACKN) output. The request line must be in the inactive state for at least one clock cycle before a request is made. After a request has been asserted, it must remain at the assertion level for at least one clock cycle. If another request is received before the first operand part of a former request is acknowledged, the second request is not recognized. Normally, the DMAI will relinquish the bus after servicing a valid request. However, if the device generates a new request before the DMAI asserts DTCN for the last operand part, the DMAI will retain ownership of the bus and that request will be serviced before the bus is relinquished.

Acknowledge (ACKN)

The DMAI asserts the acknowledge line, which implicitly addresses the device making the request, during transfers to and from the device. The line may be used to control buffering circuits between the data bus and the MPU bus.

During burst mode, REQN must not be disasserted for less than one CLK period plus four RC time constants, where R is the value of the resistor used for the pullup on BRN and C has a typical value of 20pF.

Ready (RDYN)

Ready is an active low input which is asserted by the requesting device to indicate to the DMAI that valid data has either been stored or put on the bus. If negated, it indicates that the data has not been stored or presented, causing the DMAI to enter wait states until RDYN is asserted. RDYN can be held low continuously if the device is fast enough so that wait states are not required. The current state of the ready input is reflected in CSR[8].

Done (DONEN)

Done is a bidirectional active low signal. As an output, it is asserted and negated by the DMAI concurrent with the ACKN output of the last operand part to indicate to the device that the memory transfer count is exhausted and that the DMAI's operation is completed as a result of that transfer.

The DMAI also monitors the state of the line while acknowledging a device. If the device asserts DONEN, the DMAI will terminate operation after the transfer of the current operand. In this case the DMAI clears the channel active bit and sets the channel operation complete and normal device termination bits in the CSR. If both the DMAI and the device assert DONEN, the device termination is not recognized, but the operation does terminate.

Device Transfer Complete (DTCN)

DTCN is an active low output which is asserted by the DMAI to indicate to the device that the requested data transfer is complete. On a write to memory operation, it indicates that the data provided by the device has been successfully stored. On a read from memory operation, it indicates to the device that the data from memory is present on the data bus and should be latched. DTCN is not asserted if assertion of the RERUNN input terminates the bus cycle.

Bus Arbitration

Upon receiving a valid request for a transfer from the device, the DMAI will arbitrate for and obtain ownership of the system bus.

The DMAI indicates that it wishes to become the bus master by asserting its bus request (BRN) output. This is a wire-ORed signal that indicates to the MPU that some external device requires control of the bus. The processor is effectively at a lower priority level than external devices and will relinquish the bus after it has completed the last bus cycle it has started. The processor puts the bus up for external arbitration by asserting its bus grant (BGN) output. This signal may be routed through a daisy chain (such as provided by the DMAI) or through some other priority-encoded network. When the DMAI making the bus request receives the bus grant (indicated by its BGN input being asserted), it is to be the next bus master. It waits until address strobe (ASN), data transfer acknowledge (DTACKN) and bus grant acknowledge (BGACKN) become inactive and then assumes ownership of the bus by asserting its own BGACKN output. The DMAI then negates the BRN output and proceeds with the data transfer phase. After this phase is completed, the DMAI relinquishes bus ownership by negating the BGACKN output.

In burst DMA mode, detection of an active low request input after the DMAI operation has been started will begin the bus arbitration cycle. However, if the device negates its request at least one clock cycle before the DMAI asserts BGACKN, the DMAI will negate its bus request and will not assume ownership of the bus.

Data Transfers

The actual transfer of data between the memory and the device occurs during the data transfer phase. All transfers occur during a single cycle except in the case of long word operands, in which case two cycles are used to transfer the operand as two 16-bit words. The transfers take place using a 'single address' protocol; the DMAI addresses the memory via the bus address lines, while the device is implicitly addressed via the acknowledge output.

When a request is generated using the request method programmed in the control register, the DMAI obtains the bus and asserts acknowledge to notify the device that a transfer is to take place. The DMAI asserts all S68000 bus control signals needed for the transfer and holds them until the device responds with ready. The bus cycle then terminates normally. Ready may be tied low (asserted) if the device is fast enough.

When the transfer is from memory to the device, data is valid when DTACKN is asserted by the memory and remains valid until the data strobe(s) are negated. The assertion of DTCN from the DMAI can be used to latch the data, as the data strobes are not removed until one-half clock after the assertion of DTCN.

When the transfer is from device to memory, the data must be valid on the bus before the DMAI asserts the data strobe(s). The device indicates valid data by asserting ready. The DMAI then asserts the strobes and holds them asserted until the memory accepts the data, indicated by the assertion of DTACKN. The DMAI then asserts DTCN and negates the data strobes.

Flow charts for these operations are shown in figures 1 and 2. Refer to the timing section for the equivalent timing diagrams.

Operation Termination

Termination of the block transfer occurs under the conditions detailed below.

Terminal Count

As part of each transfer of an operand, the DMAI decrements the memory transfer counter. If this counter is decremented to zero, the operand is the last operand of the block. The DMAI operation is complete and it notifies the device of completion by asserting the DONEN output during the last operand transfer cycle. When the transfer has been completed, the channel active bit in the CSR is cleared and the COC bit is set, unless an error occurs.

Device Termination

The DMAI monitors the state of the DONEN line while acknowledging a device transfer request. If the device asserts DONEN, the DMAI will terminate operation after the transfer of the current operand. When the transfer has been completed, the DMAI clears the' channel active bit and sets the COC and normal device termination bits in the CSR. If both the DMAI and the device assert DONEN, the device termination is not recognized, but the operation does terminate.

Software Abort

The software abort bit (CCR[4]) allows the MPU to abort the current operation of the DMAI. The COC and error bits in the CSR are set, the channel active bit in the CSR is cleared, and an abort error condition is signaled in the CER.

Rerun Error

The DMAI provides a rerun input (RERUNN) to indicate a bus exception condition. RERUNN must arrive prior to or in coincidence with DTACKN in order to be recognized, and the DMAI verifies that the line has been stable for two clock cycles before acting on it. The occurrence of a rerun during a DMAI bus cycle forces it to terminate the bus cycle in an orderly manner.

When the assertion of rerun is verified, the DMAI stops operation and three-states the data, address, and control lines, except BGACKN, so that it retains ownership of the bus. It remains halted until rerun becomes inactive, and then re-tries the last bus cycle. If rerun is asserted again, the DMAI stops DMA operation, releases the bus, sets the error and COC bits in the CSR, clears the active bit in the CSR, and sets the error code in the CER to indicate a bus error.

While stopped due to assertion of rerun, the DMAI does not generate any bus cycles and will not honor any requests until it is removed. However, the DMAI still recognizes requests.

Error Recovery Procedure

If an error occurs during a DMA transfer such that the DMAI stops the DMA operation, information is available to the operating system for an error recovery routine.

The information available to the operating system consists of the memory address counter, the memory transfer counter, and the control, status, and error registers. The DMAI decrements the memory transfer counter before attempting a DMA operation, so the register will contain the count minus one of the attempted transfer. The memory address counter will contain the address at which the DMA operation was attempted.

Reset

The reset input (RESETN) provides a means of resetting and initializing the DMAI from an external source. If the DMAI is a bus master when reset is received, the DMAI relinquishes the bus. Reset clears the control and error registers, sets all bits of the status register except CSR[8] to zero, and initializes the interrupt vector register to H'0'F.

Interrupts

The interrupt enable bit (CCR[3]) determines whether the DMAI generates interrupt requests. When the bit is set, an interrupt request is generated if the channel operation complete bit in the CSR is set. When the IACKN input is asserted, and the DMAI has an interrupt request pending, the DMAI returns an interrupt vector on the data bus.

The interrupt vector issued is the contents of the IVR. Only the seven most significant bits of the programmed value are used by the DMAI. The vector from the DMAI contains a zero in the LSB position if a normal termination occurred (error bit not set) and contains a one in the LSB position if termination was due to an error (error bit set).

The contents of this register are initialized to H'0F' by a reset. The value returned will be H'0F', regardless of the error state, until the register is programmed by the MPU.

To provide compatibility with the other 68K family DMA controllers, the IVR has two addresses (see table 1). If program compatibility is required, the value written at the

Direct Memory Access Interface (DMAI) SCB68430

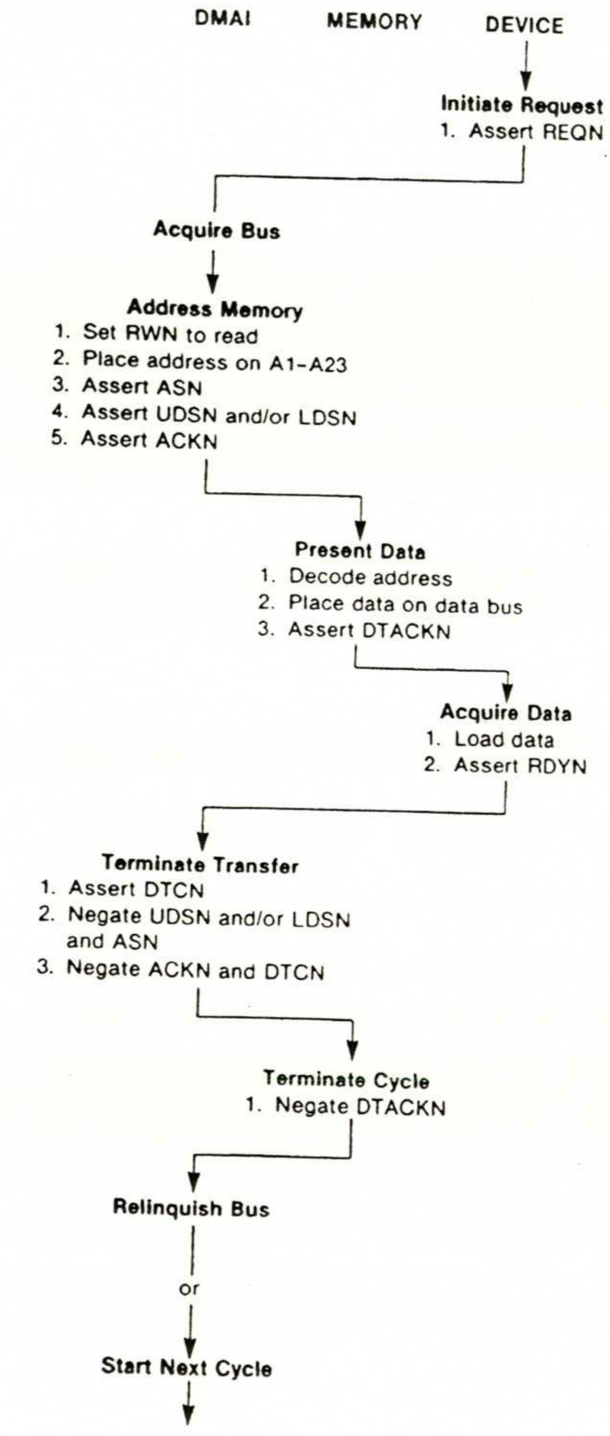


Figure 1. Transfer from Memory to Device Flowchart

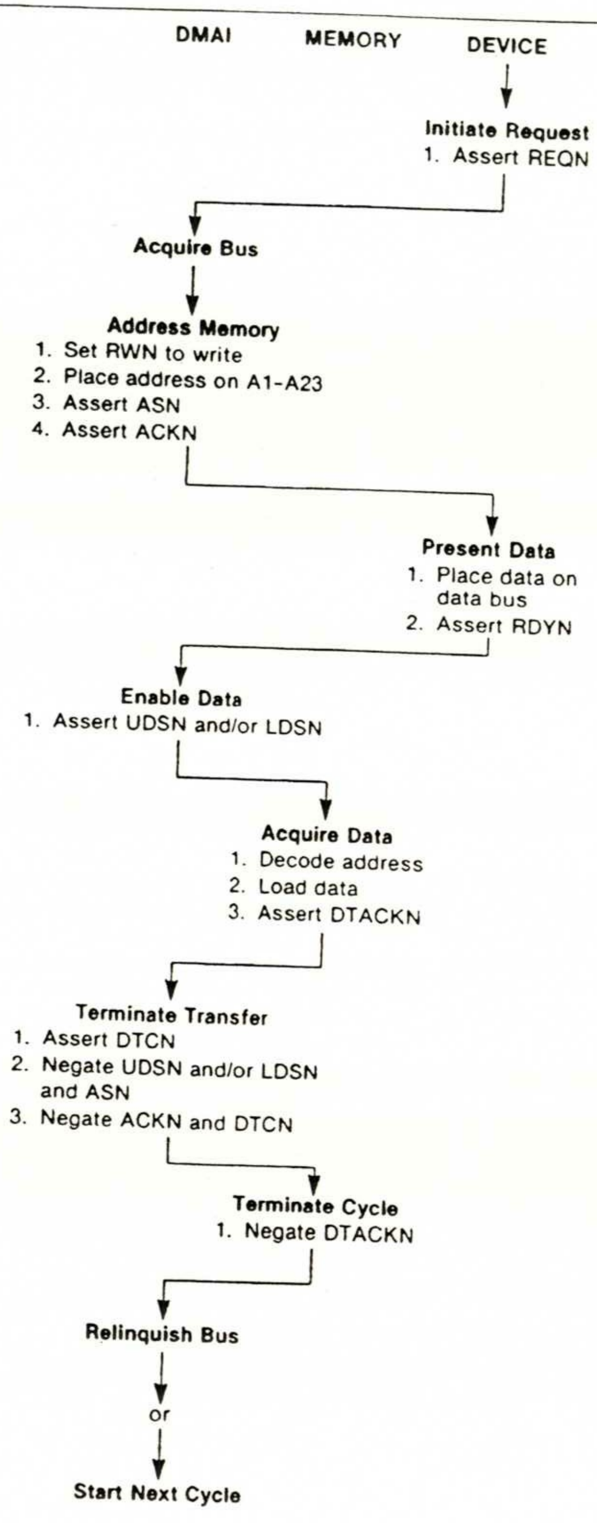


Figure 2. Transfer from Device to Memory Flowchart

normal IVR address should have a zero as its LSB, and the value written at the error IVR address should be the same but with the LSB equal to one.

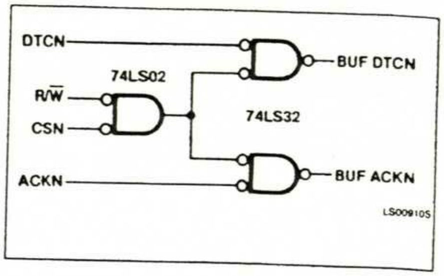
APPLICATIONS

Figure 3 illustrates a typical interconnection of the DMAI in a 68000 based system.

DESIGN NOTE

When clearing the error bit in CSR (bit 12) after a DMAI abort due to a double RERUNN, ACKN and DTCN will both go low concurrent with CSN and DTACKN for one CLK cycle.

To prevent the possibility that the device may misinterpret these signals, it is suggested that these signals be ANDed with CSN (see figure below).



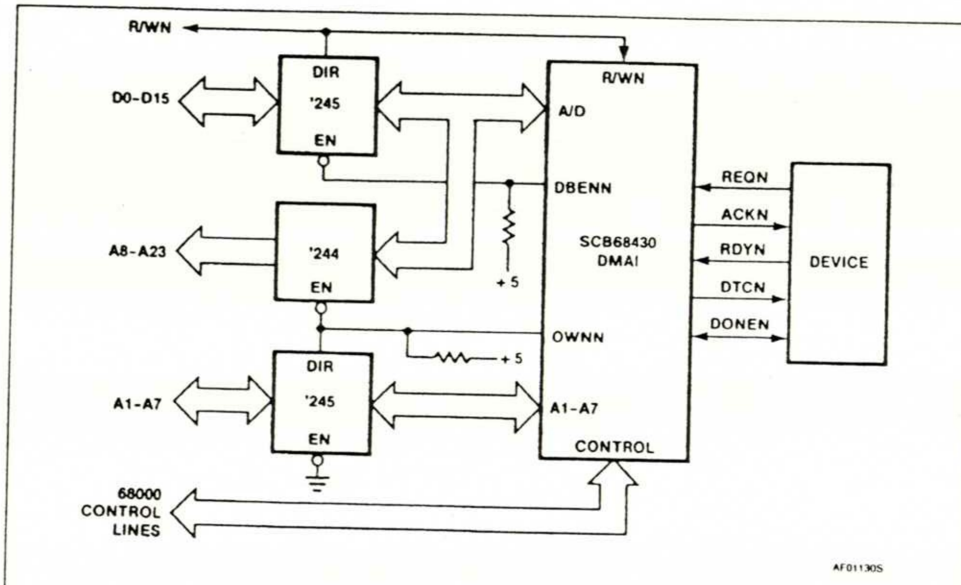


Figure 3. DMAI Application

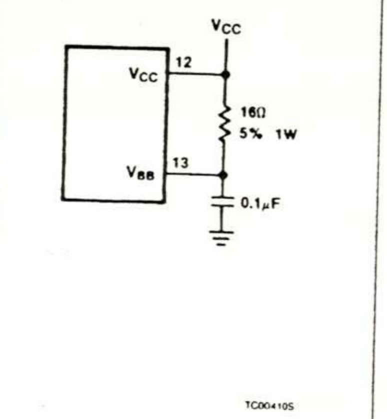


Figure 4. Recommended V_{BB} Test Circuit

Direct Memory Access Interface (DMAI) SCB68430

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Supply voltages V_{CC} and V_{BB}	−0.5 to +7.0	V
Input voltage	−0.5 to +5.5	V
Operating temperature range[2]	0 to +70	°C
Storage temperature	−65 to +150	°C

DC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0V ±5%, V_{BB} = Figure 4, T_A = 0°C to +70°C[3,4,7]

	PARAMETER	TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
V_{IL}	Input low voltage			0.8	V
V_{IH}	Input high voltage		2.0		V
V_{OL}	Output low voltage	I_{OUT} = 5.3mA		0.5	V
V_{OH}	Output high voltage, all outputs except open collector outputs[5]	I_{OUT} = −400μA	2.5		V
I_{IL}	Input low current	V_{IN} = 0.4V		−400	μA
I_{IH}	Input high current	V_{IN} = 2.7V		20	μA
I_{OC}	Open collector off state current[5]	V_{OUT} = 2.4V		20	μA
I_{SC}	Output short circuit current[6]	V_{CC} = max	−40	−100	mA
I_{CC}	V_{CC} supply current	V_{CC} = max		130	mA
I_{BB}	V_{BB} supply current			275	mA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is stress rating only and functional operation of the device at these or at any other conditions other than those indicated in the Electrical Characteristics section of this data sheet is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. Parameters are valid over specified temperature range.
4. All voltage measurements are referenced to ground (V_{SS}). For testing, all signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V as appropriate.
5. IRQN, BRN, DONEN, and OWNN are open collector outputs.
6. No more than one output should be connected to ground at one time.
7. Capacitive test load is 100pF for all pins except DTCN which has a 35pF capacitive test load.

AC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0V ±5%, V_{BB} = Figure 4, T_A = 0°C to +70°C[3,4,7]

NO.	FIGURE	CHARACTERISTICS	TENTATIVE LIMITS				UNIT
			10MHz		12.5MHz		
			Min	Max	Min	Max	
1	5	A1 – A7, ASN, RWN, set-up to UDSN, LDSN low	0		0		ns
2	5	D0 – D15 3-state to invalid data from ASN, CSN, and UDSN or LDSN low	10		10		ns
3	5	DTACKN 3-state to high from ASN, CSN, and UDSN or LDSN low	10		10		ns
4	5	CSN low after UDSN or LDSN low		25		25	ns
5	5, 6	DBENN low after ASN and CSN low		60		60	ns
6	5	D0 – D15 valid data from ASN, CSN, and UDSN or LDSN low		100		100	ns
7	5	DTACKN low after D0 – D15 valid data	−15	30	−15	30	ns
8	5	A1 – A7, ASN, RWN or CSN hold after UDSN and LDSN high	0		0		ns
9	5, 6	DBENN high from either ASN or CSN high		45		45	ns
10	5	D0 – D15 to 3-state from UDSN and LDSN high		80		80	ns
11	5	D0 – D15 to invalid data from UDSN and LDSN high	10		10		ns
12	5, 6	DTACKN high from UDSN and LDSN high		55		55	ns
13	5, 6	DTACK 3-state from either CSN or ASN high		85		85	ns
14	6	A1 – A7, ASN, RWN set-up to UDSN, LDSN low	50		50		ns/s
15	6	CSN set-up before UDSN or LDSN low	20		20		ns
16	6	DTACKN 3-state to high after CSN and ASN low	10		10		ns
17	6	D0 – D15 valid after UDSN or LDSN low		0		0	ns
18	6	DTACKN low from UDSN or LDSN low		100		100	ns
19	6	UDSN and LDSN low time	115		100		ns
20	6	A1 – A7 hold after UDSN and LDSN high	0		0		ns
21	6	ASN, RWN and CSN hold after UDSN and LDSN high	0		0		ns

Direct Memory Access Interface (DMAI) SCB68430

AC ELECTRICAL CHARACTERISTICS (Continued)

NO.	FIGURE	CHARACTERISTICS	TENTATIVE LIMITS				UNIT
			10MHz		12.5MHz		
			Min	Max	Min	Max	
22	6	D0 – D15 hold after UDSN and LDSN high	0		0		ns
23	7	DBENN low from last low of ASN, IACKN, LDSN		65		65	ns
24	7	D0 – D7 valid after last low of ASN, IACKN, LDSN		105		105	ns
25	7	DTACKN 3-state to high after last low of ASN, IACKN, LDSN		100		100	ns
26	7	DTACKN low after last low of ASN, IACKN, LDSN		110		110	ns
27	7	DBENN high after first high of ASN, IACKN, LDSN		55		55	ns
28	7	D0 – D7 hold after first high of ASN, IACKN, LDSN		60		60	ns
29	7	D0 – D7 3-state after first high of ASN, IACKN, LDSN		80		80	ns
30	7	DTACKN high after first high of ASN, IACKN, LDSN		60		60	ns
31	7	DTACKN 3-state after first high of ASN, IACKN, LDSN		95		95	ns
32	8	BRN high from CLK high		65		65	ns
33	8, 11, 12	BGACKN low from CLK low		75		75	ns
34	8, 11, 12	OWNN low from CLK high		75		75	ns
35	8	BGACKN high from CLK low		75		75	ns
36	8,11,12	OWNN high from CLK high (load dependent)		50		50	ns
37	10	REQN set-up before CLK low	30		30		ns
38	10	REQN hold after CLK high	20		20		ns
39	10	BRN low from CLK high		80		80	ns
41	11, 12	ASN, UDSN, LDSN, RWN 3-state to high from CLK low		75		75	ns
43	11, 12	A1 – A23 to valid ASN	0		0		ns
44	11, 12	ASN low from CLK high		65		65	ns
45	11, 12	LDSN, UDSN low from CLK high		90		90	ns
46	11, 12	ACKN low from CLK high		65		65	ns
47	11, 12	DTACKN set-up to CLK high	30		30		ns
48	11, 12	RDYN set-up to CLK low	30		30		ns
49	11, 12	DTCN low from CLK high		70		70	ns
50	11, 12	ASN high from CLK high		75		75	ns
51	11, 12	LDSN, UDSN, high from CLK high		90		90	ns
52	11, 12	DTACKN, RDYN hold after CLK high	0		0		ns
–	11, 12	ASN, LDSN, UDSN, high from DTCN low	–20		–20		ns
53	11, 12	ACKN high from CLK high		50		50	ns
54	11, 12	DTCN high from CLK high		50		50	ns
55	11, 12	Address valid after CLK low	10		10		ns
–	11, 12	Address valid after ASN high	0		10		ns
56	11, 12	DONEN (output) low from CLK low		120		120	ns
57	11, 12	DONEN (output) high from CLK high		50		50	ns
58	11, 12	DONEN (input) set-up low before CLK low	30		30		ns
59	11, 12	DONEN (input) hold low after CLK high	0		0		ns
60	11, 12	BGACKN, ASN, UDSN, LDSN, RWN to 3-state from CLK low		75		75	ns
62	11, 12	A1 – A23 valid to 3-state from CLK high		100		100	ns
63	12	R/WN low from CLK high		65		65	ns
64	12	R/WN high from CLK high		75		75	ns
65	13	RERUNN set-up low before CLK high	30		30		ns
66	13	RERUNN hold low from CLK high	20		20		ns
67	13	A1 – A23 to idle state from CLK low		100		100	ns
68	13	A1 – A23 to valid after CLK low		85		85	ns

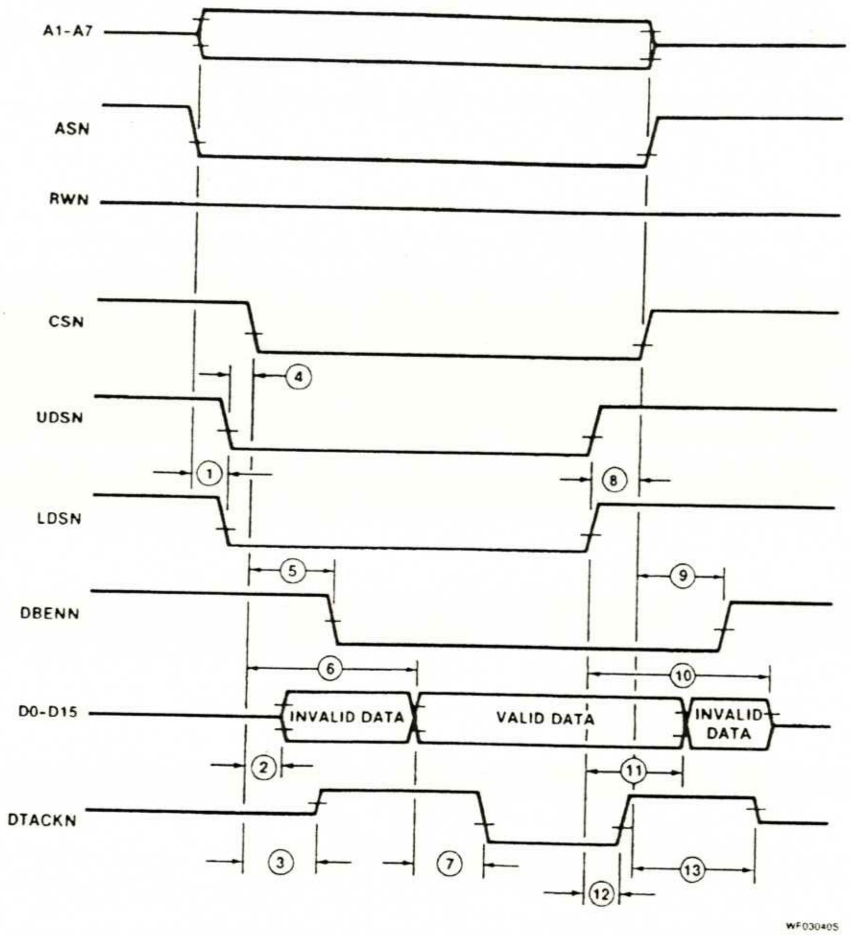


Figure 5. DMAI Read Timing

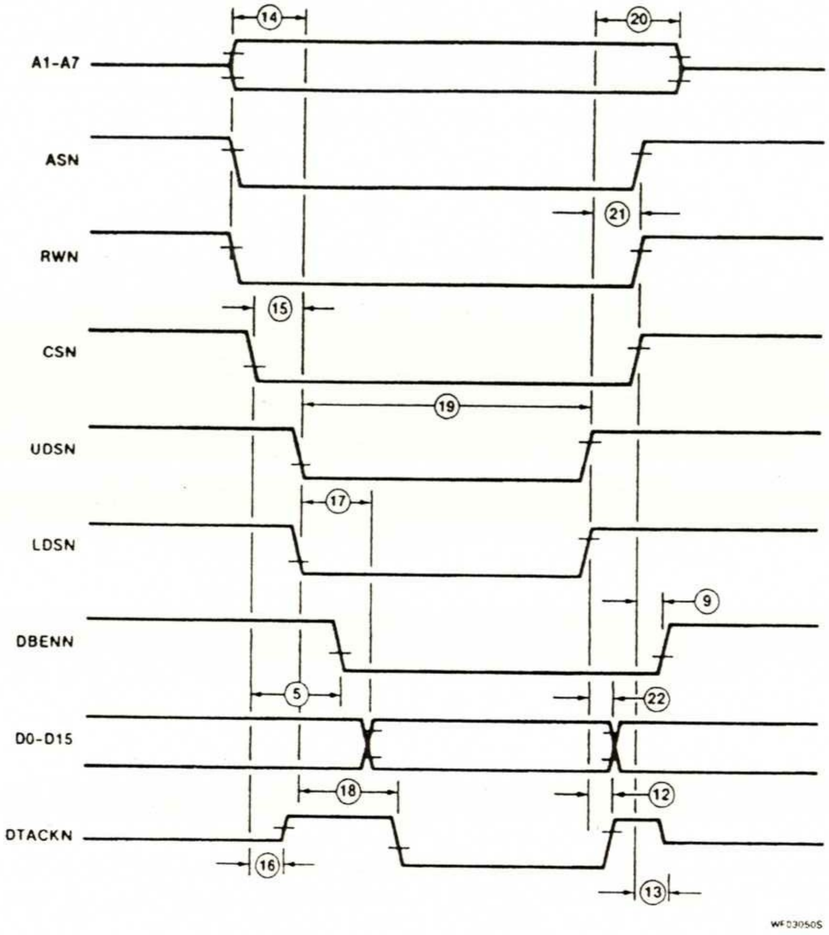


Figure 6. DMAI Write Timing

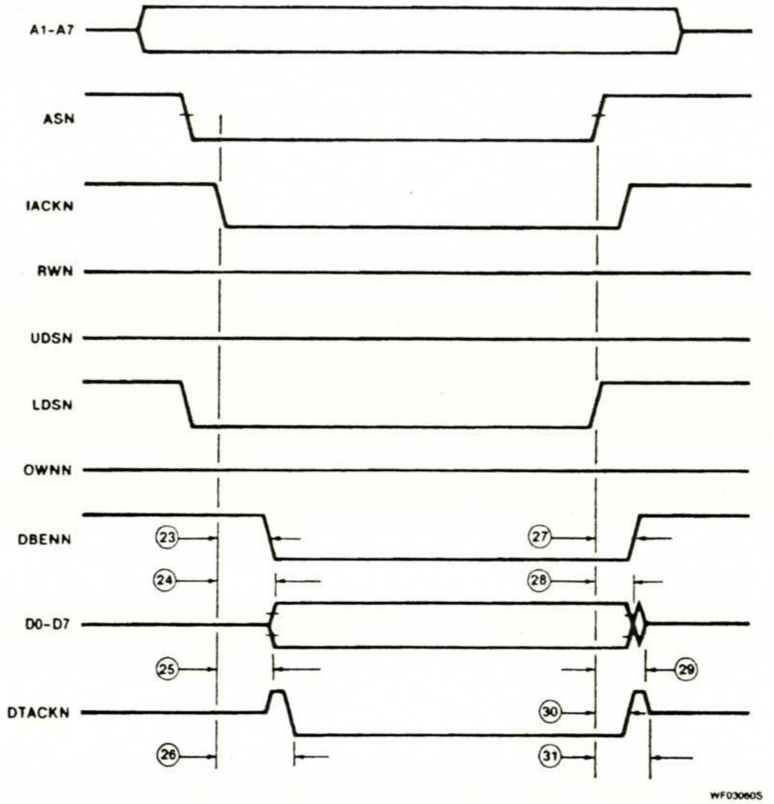
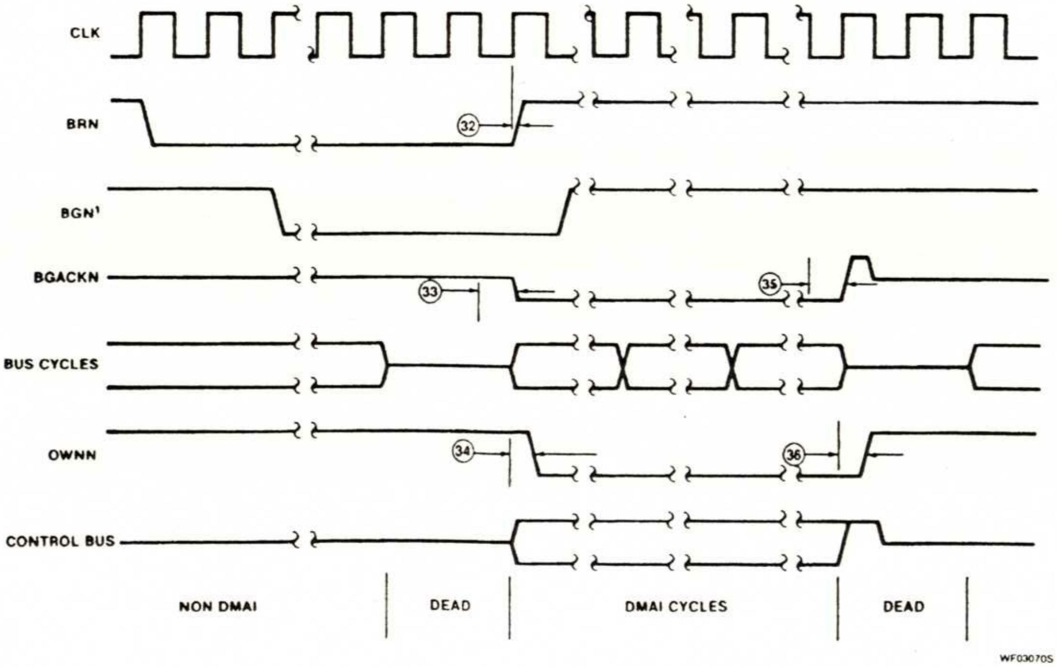
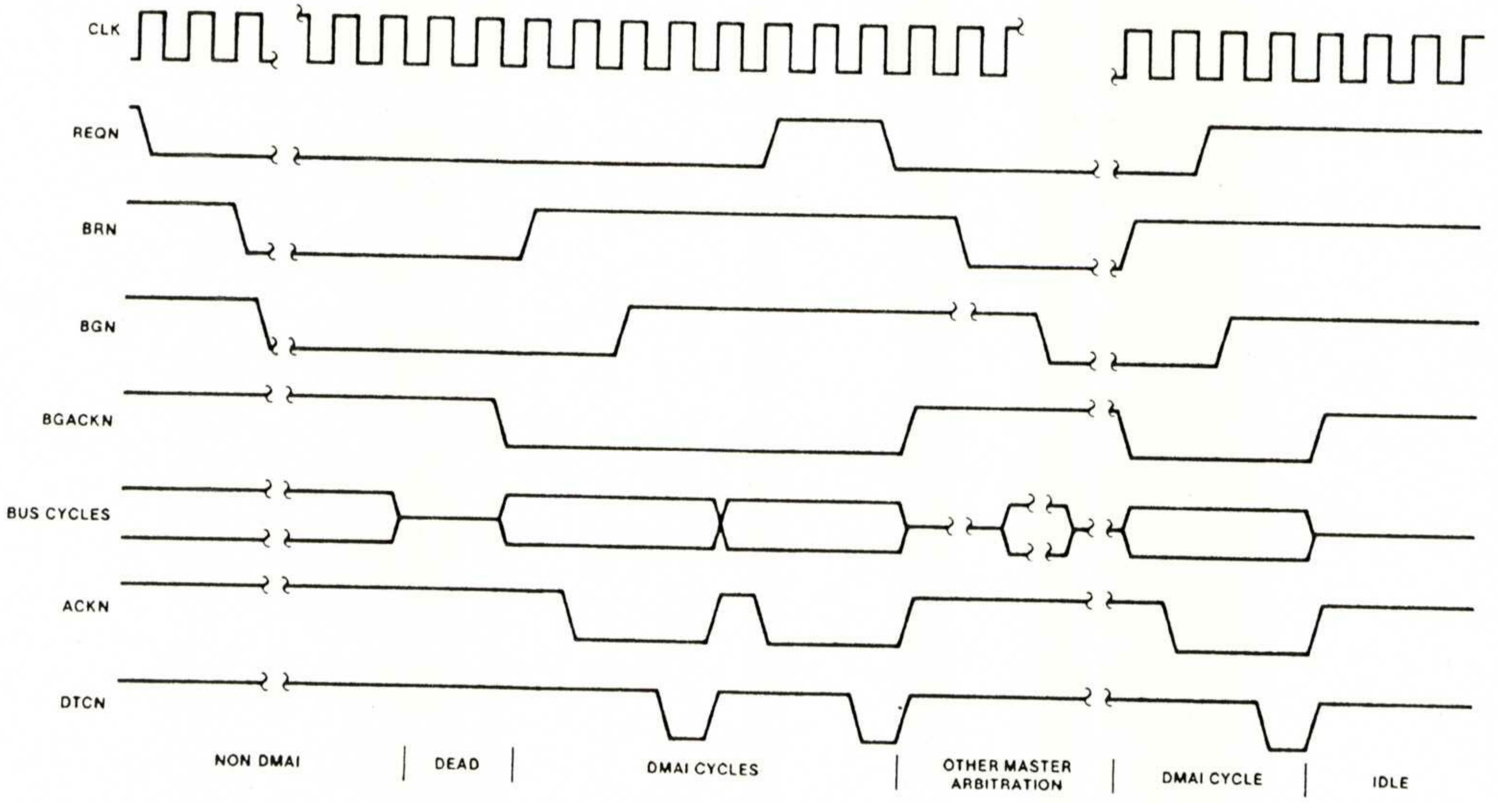


Figure 7. CPU IACK Cycle to DMAI



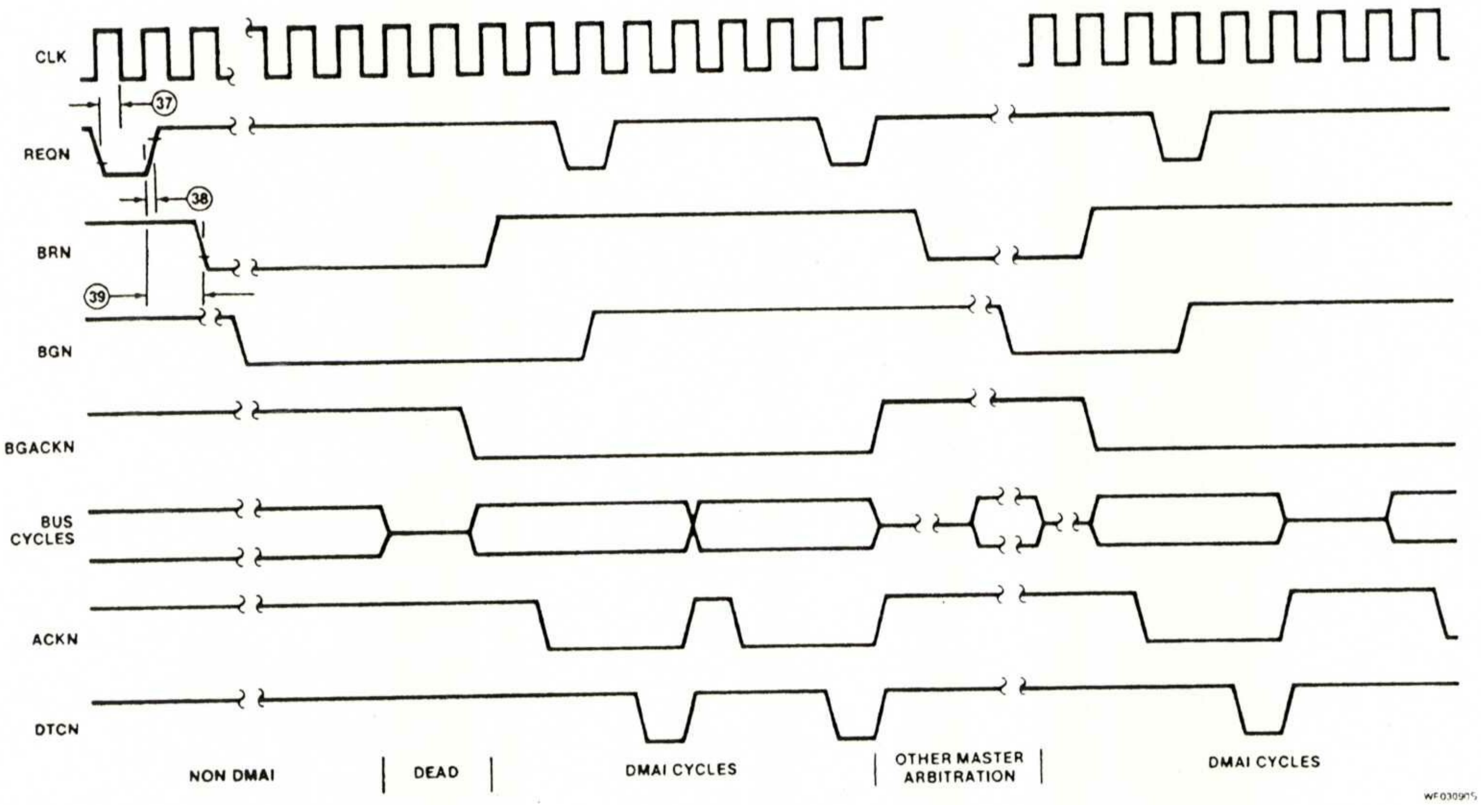
NOTES:
1. Device will become master if BGN is asserted concurrent with or later than REQN (same clock edge or later).
2. ASN, DTACKN and BGACKN must be negated in order for DMAI to become master. Timing assumes all these happen concurrent with BGN — if not, it is from the latest signal which is negated.

Figure 8. DMAI Bus Arbitration Timing



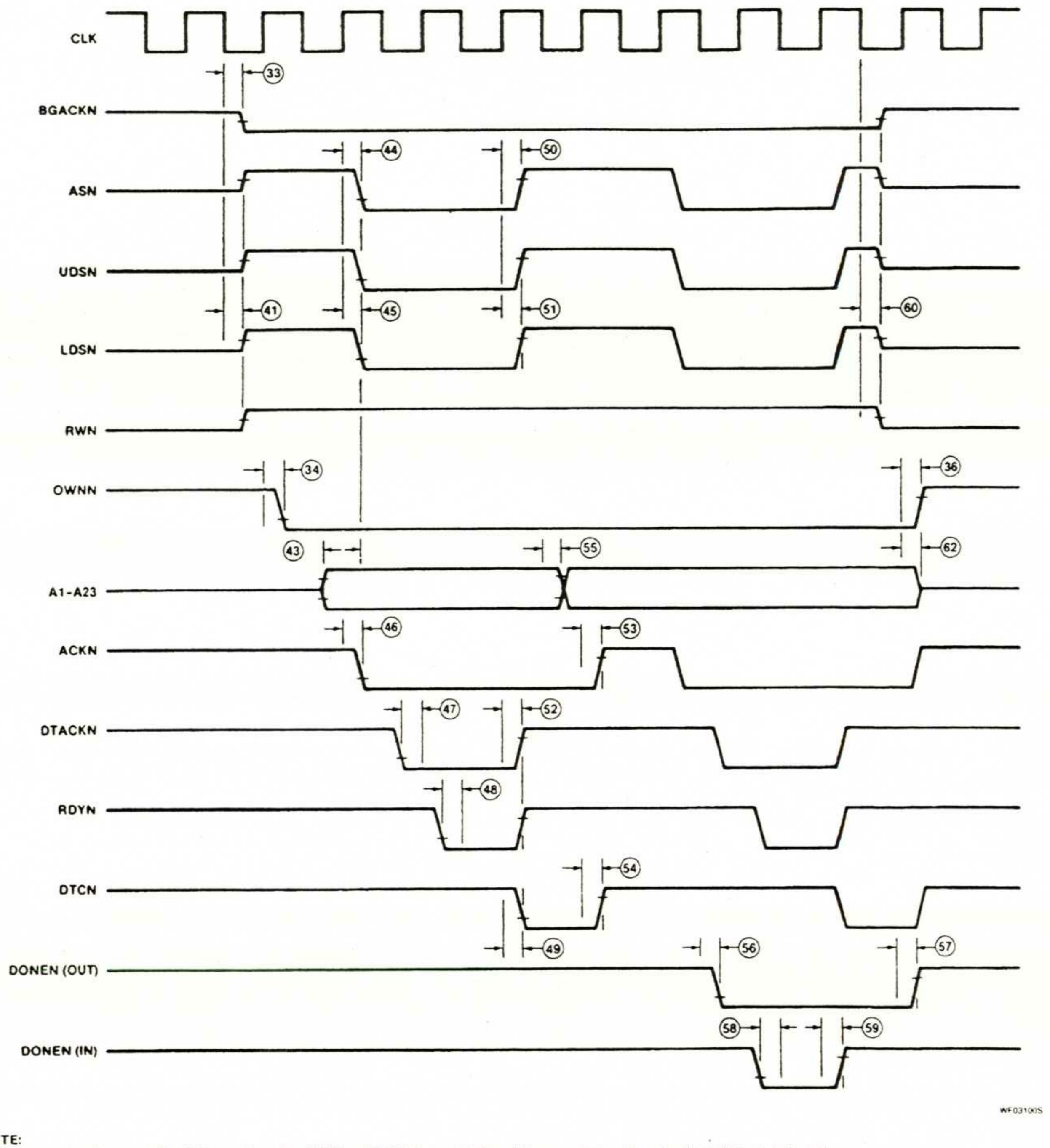
NOTES:
1. To maintain mastership next bus cycle, REQN must remain asserted at least until 1/2 clock cycle after DTCN is asserted.
2. To guarantee that mastership is relinquished next cycle, REQN must be negated no later than 1/2 clock cycle prior to DTCN.

Figure 9. DMAI Burst Mode Request Timing



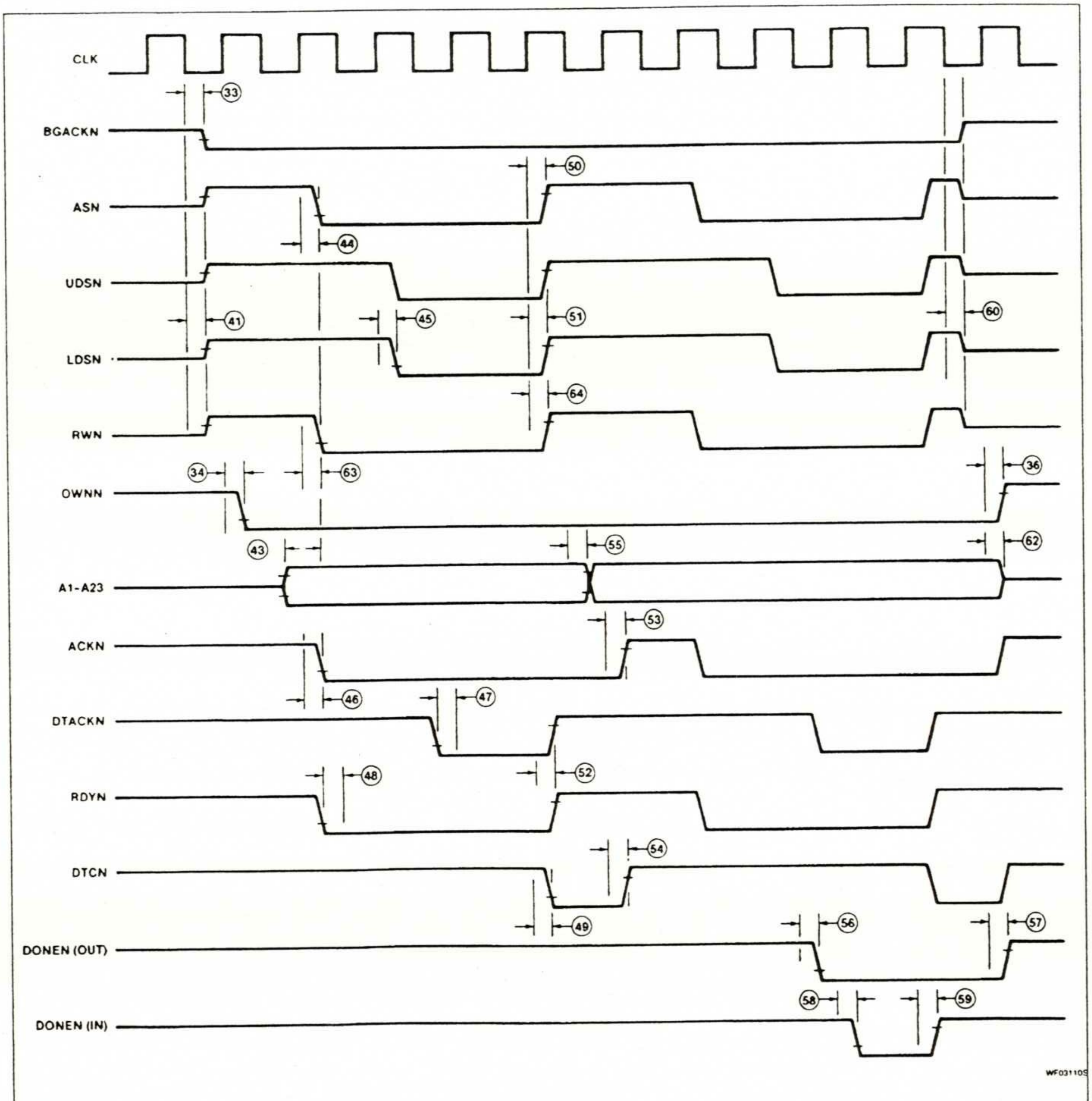
1. In order to keep the bus, REQN must come no later than the 1/2 clock minus the set-up time ㊲ prior to assertion edge of DTCN.

Figure 10. DMAI Cycle Steal Mode Request Timing



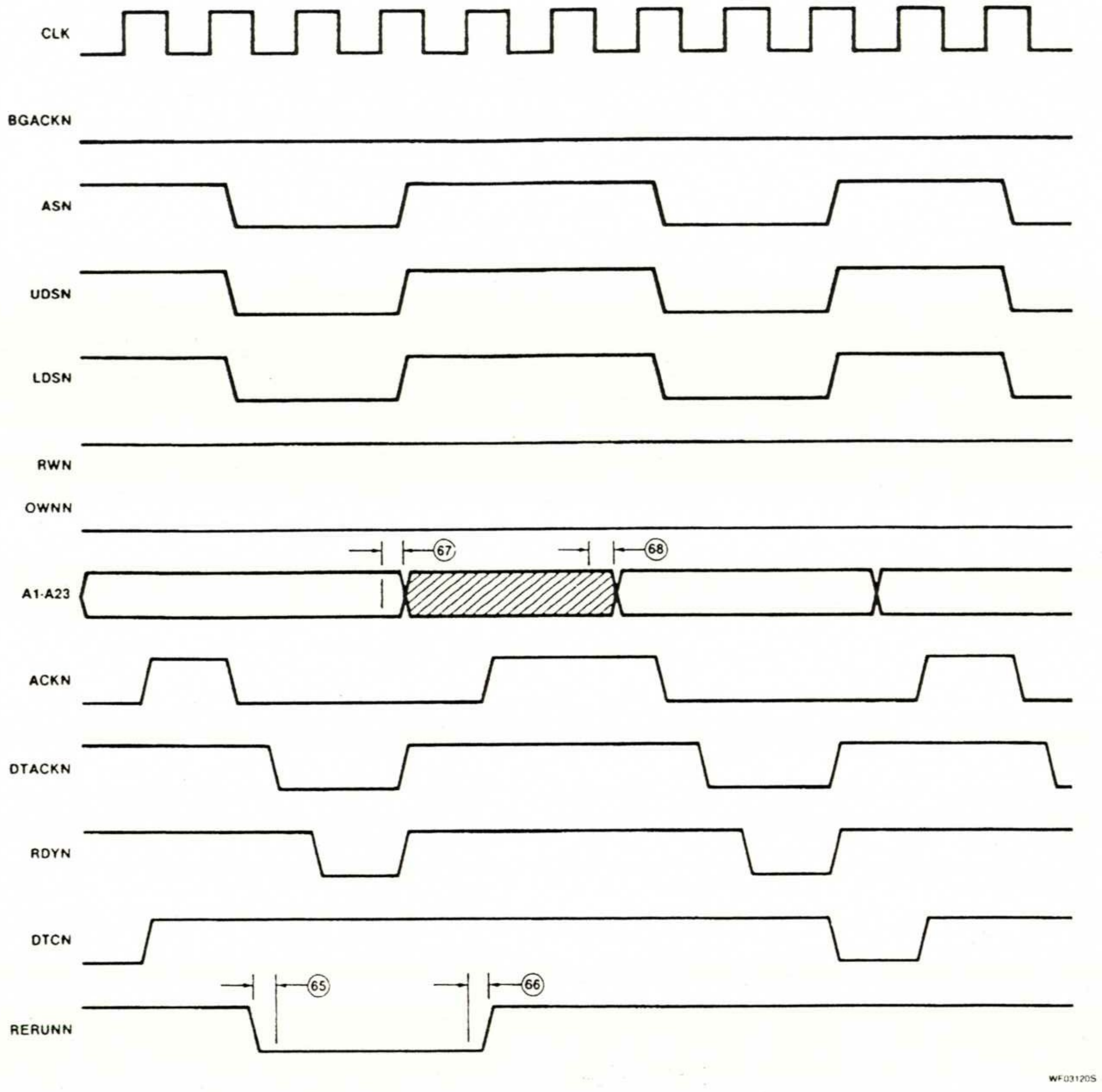
NOTE:
1. 16-bit transfer illustrated. For 8-bit transfer either LDSN or UDSN, but not both, will be asserted each cycle, depending on byte address.

Figure 11. Read from Memory, Write to Device



NOTE:
1. 16-bit transfer illustrated. For 8-bit transfer either LDSN or UDSN, but not both, will be asserted each cycle, depending on byte address.

Figure 12. Write to Memory, Read from Device



NOTES:

1. 16-bit transfer illustrated. For 8-bit transfer either LDSN or UDSN, but not both, will be asserted each cycle, depending on byte address.
2. DMAI will release the bus after a RERUNN if there is no valid request. The next request will then retry the cycle which was terminated by the RERUNN signal.
3. RERUNN must be asserted no later than DTACKN and RDYN.
4. If a cycle is terminated by RERUNN, the transfer count will be one less than the actual data transferred correctly. The double RERUNN signal on the same cycle will terminate the DMAI operation with a status bit set and an interrupt generated (if enabled).

Figure 13. Rerun Asserted During Read from Memory, Write to Device

APPENDIX C

SAMPLE SOFTWARE LISTINGS

src=dr11wa4.sa

```
******************************************************************
*                                                                *
*       DR11A4:   DR11WA DATA TRANSFER TESTS                     *
*       ---------------------------------------------------      *
*       MODE_M:   Start test processing sequence as a master.    *
*       MODE_S:   Start test processing sequence as a slave.     *
*                                                                *
******************************************************************

*       This is the main entry point to transfer tests

        XDEF      DR11A4                drllwA Dma (+) transfer tests

*       These entry points are for setup modifications

        XDEF      SETMTC                set the transfer counter
        XDEF      SETMAC                set the address counter
        XDEF      SETAMR                set the address modifier
        XDEF      SETEMA                set the extended address
        XDEF      SETC33                set def's for cpu33 buffers
        XDEF      SETRR2                set def's for rr2 buffers
        XDEF      DUMP_T                dump dma transfer tables
        XDEF      TEST                  transfer test enable flags
        XDEF      TDATA                 reference single test data

*       Reference external subroutines and variables

        XREF      DSPBYTE               display byte subr
        XREF      DSPWORD               display word subr
        XREF      DSPLONG               display long subr
        XREF      FLAGS                 program (test) flags
        XREF      PCOUNT                loop on test pass count
        XREF      FCOUNT                loop on test fail count

        INCLUDE   VMEPROM.SA
******************************************************************
*                                                                *
*             VMEPROM INTERFACE SUPPORT CODE                     *
*       =================================================        *
*       Copyright 1989 Vme Microsystems International Corp.      *
*       =================================================        *
*       VMEPROM.SA  contains  the  assembler  options,           *
*       constants  and macros  for  assembling  Development      *
*       Test  code that is to be ran on a Force  Cpu  board      *
*       under  Vmeprom/Pdos. This code is  presently  being      *
*       assembled on the motorola Vme/10 development system      *
*       or  compatable  cross-assembler  on  the  IBM   PC       *
*       (msdos).  This code is included in the test  source      *
*       at assembly time in each module as it is needed. It      *
*       is usually listed only once at the beginning in the      *
*       primary or 'main' module of the listing print file.      *
*       Recently included  are parameters for four commonly      *
*       used terminals and the macro to install them in the      *
*       TCB of the current task.  (this must be done before      *
                       *        changing register A6 in the test code software).           *
                       *        ---------------------------------------------------        *
                       *        Version 2.00.02.14.90    by   Clark Coe Gwathney Jr.       *
                       *                                                                   *
                       **********************************************************************
                       *
                       *******[ TYPICAL ASSEMBLER OPTIONS ]**********************************
                       *
                                OPT       P=68020             use 68020 instruction coding
                                OPT       NOCEX               do not print dc.x definitions
                                OPT       NOMEX               do not print macro expansions
                       *
                       *******[ ASCII CONTROL EQUATES ]**************************************
                       *
      <00000000>       NUL      EQU       00                  null
      <00000001>       SOH      EQU       01                  start of header
      <00000002>       STX      EQU       02                  start of text
      <00000003>       ETX      EQU       03                  end of text
      <00000004>       EOT      EQU       04                  end of transmission
      <00000005>       ENQ      EQU       05                  inquiry
      <00000006>       ACK      EQU       06                  acknowledge
      <00000007>       BEL      EQU       07                  bell (beep)
      <00000008>       BS       EQU       08                  backspace
      <00000009>       HT       EQU       09                  horizontal tab
      <0000000A>       LF       EQU       10                  line feed
      <0000000B>       VT       EQU       11                  vertical tab
      <0000000C>       FF       EQU       12                  form feed
      <0000000D>       CR       EQU       13                  carraige return
      <0000000E>       SO       EQU       14
      <0000000F>       SI       EQU       15
      <00000010>       DLE      EQU       16
      <00000011>       DC1      EQU       17                  device control 1 (xon)
      <00000012>       DC2      EQU       18                  device control 2
      <00000013>       DC3      EQU       19                  device control 3 (xoff)
      <00000014>       DC4      EQU       20                  device control 4
      <00000015>       NAK      EQU       21                  not acknowledge
      <00000016>       SYN      EQU       22
      <00000017>       ETB      EQU       23                  end of transmission block
      <00000018>       CAN      EQU       24                  cancel
      <00000019>       EM       EQU       25
      <0000001A>       SUB      EQU       26
      <0000001B>       ESC      EQU       27                  escape
      <0000001C>       FS       EQU       28
      <0000001D>       GS       EQU       29
      <0000001E>       RS       EQU       30
      <0000001F>       US       EQU       31
      <0000007F>       DEL      EQU       127                 delete
                       *
                       *******[ VMEPROM / PDOS SYSTEM CALL EQUATES ]*************************
                       *
      <0000A006>       X881     EQU       $A006               save 68881 flags on switch
      <0000A0AA>       XAPF     EQU       $A0AA               append file to file
      <0000A070>       XBCP     EQU       $A070               set baud on console port
      <0000A072>       XCBC     EQU       $A072               check for break character
      <0000A050>       XCBD     EQU       $A050               convert binary to decimal
```

```
<0000A052>        XCBH     EQU        $A052              convert binary to hex
<0000A054>        XCBM     EQU        $A054              convert to decimal w/message
<0000A074>        XCBP     EQU        $A074              check for break or pause
<0000A06A>        XCBX     EQU        $A06A              convert to decimal in buffer
<0000A056>        XCDB     EQU        $A056              convert ascii to binary
<0000A0D0>        XCFA     EQU        $A0D0              close file with attribute
<0000A068>        XCHX     EQU        $A068              convert binary to hex in buffer
<0000A0D2>        XCLF     EQU        $A0D2              close opened file
<0000A076>        XCLS     EQU        $A076              clear screen (TCB parms)
<0000A0AE>        XCPY     EQU        $A0AE              copy file
<0000A026>        XCTB     EQU        $A026              create task block
<0000A032>        XDEV     EQU        $A032              delay set/reset event
<0000A0D4>        XDFL     EQU        $A0D4              define file
<0000A0D6>        XDLF     EQU        $A0D6              delete file
<0000A04A>        XDMP     EQU        $A04A              dump memory from stack
<0000A114>        XDPE     EQU        $A114              delay physical event
<0000A024>        XDTV     EQU        $A024              define trap vectors
<0000A00C>        XERR     EQU        $A00C              return error to vmeprom
<0000A030>        XEXC     EQU        $A030              execute pdos call in d7.w
<0000A00E>        XEXT     EQU        $A00E              exit to vmeprom
<0000A0CE>        XFAC     EQU        $A0CE              file altered check
<0000A0F8>        XFBF     EQU        $A0F8              flush buffers
<0000A0A0>        XFFN     EQU        $A0A0              fix file name
<0000A058>        XFTD     EQU        $A058              fix time & date
<0000A040>        XFUM     EQU        $A040              free user memory
<0000A048>        XGCB     EQU        $A048              conditional get char
<0000A078>        XGCC     EQU        $A078              get char conditional
<0000A09E>        XGCP     EQU        $A09E              get char from port
<0000A07A>        XGCR     EQU        $A07A              get character raw
<0000A07C>        XGLB     EQU        $A07C              get line in buffer
<0000A07E>        XGLM     EQU        $A07E              get line in mon. buff
<0000A080>        XGLU     EQU        $A080              get line in user buff
<0000A010>        XGML     EQU        $A010              get memory limits
<0000A004>        XGMP     EQU        $A004              get message pointer
<0000A05A>        XGNP     EQU        $A05A              get next parameter
<0000A01E>        XGTM     EQU        $A01E              get task message
<0000A03E>        XGUM     EQU        $A03E              get user memory
<0000A0C0>        XISE     EQU        $A0C0              initialize sector
<0000A0FA>        XKTB     EQU        $A0FA              kill task
<0000A028>        XKTM     EQU        $A028              kill task message
<0000A0B0>        XLDF     EQU        $A0B0              load file
<0000A03A>        XLER     EQU        $A03A              load error register
<0000A0A2>        XLFN     EQU        $A0A2              look for file name
<0000A0D8>        XLKF     EQU        $A0D8              lock file
<0000A014>        XLKT     EQU        $A014              lock task
<0000A02E>        XLSR     EQU        $A02E              load status register
<0000A0DA>        XNOP     EQU        $A0DA              open random shared file
<0000A00A>        XPAD     EQU        $A00A              pack ascii date
<0000A084>        XPBC     EQU        $A084              put buffer to console
<0000A086>        XPCC     EQU        $A086              put char's to console
<0000A088>        XPCL     EQU        $A088              put cr,lf to console
<0000A0BC>        XPCP     EQU        $A0BC              place char in port buffer
<0000A0BA>        XPCR     EQU        $A0BA              put character raw
<0000A096>        XPDC     EQU        $A096              put data to console
<0000A09C>        XPEM     EQU        $A09C              put encoded msg to console
```

```
    <0000A08A>      XPLC     EQU      $A08A             put line to console
    <0000A08C>      XPMC     EQU      $A08C             put message to console
    <0000A08E>      XPSC     EQU      $A08E             put cursor @ row,col
    <0000A0DC>      XPSF     EQU      $A0DC             position file
    <0000A098>      XPSP     EQU      $A098             put space to console
    <0000A0DE>      XRBF     EQU      $A0DE             read bytes from file
    <0000A0B2>      XRCN     EQU      $A0B2             reset console inputs
    <0000A092>      XRCP     EQU      $A092             read port cursor position
    <0000A0A6>      XRDE     EQU      $A0A6             read next directory entry
    <0000A02A>      XRDM     EQU      $A02A             dump processor registers
    <0000A0A8>      XRDN     EQU      $A0A8             read dir entry by name
    <0000A05C>      XRDT     EQU      $A05C             read date
    <0000A0E0>      XRFA     EQU      $A0E0             read file attributes
    <0000A0FE>      XRFP     EQU      $A0FE             read file position
    <0000A0E2>      XRLF     EQU      $A0E2             read line from file
    <0000A0E4>      XRNF     EQU      $A0E4             rename file
    <0000A0E6>      XROO     EQU      $A0E6             open random read only file
    <0000A0E8>      XROP     EQU      $A0E8             open random file
    <0000A094>      XRPS     EQU      $A094             read port status
    <0000A0C2>      XRSE     EQU      $A0C2             read sector
    <0000A042>      XRSR     EQU      $A042             read status register
    <0000A0B4>      XRST     EQU      $A0B4             reset disk
    <0000A0C4>      XRSZ     EQU      $A0C4             read sector zero
    <0000A044>      XRTE     EQU      $A044             return from interrupt
    <0000A05E>      XRTM     EQU      $A05E             read time
    <0000A034>      XRTP     EQU      $A034             read time parameters
    <0000A012>      XRTS     EQU      $A012             read task status
    <0000A0EA>      XRWF     EQU      $A0EA             rewind file
    <0000A018>      XSEF     EQU      $A018             set event flag w/swap
    <0000A046>      XSEV     EQU      $A046             set event flag
    <0000A002>      XSMP     EQU      $A002             send message pointer
    <0000A112>      XSOE     EQU      $A112             suspend on physical event
    <0000A0EC>      XSOP     EQU      $A0EC             open sequential file
    <0000A09A>      XSPF     EQU      $A09A             set port flag
    <0000A020>      XSTM     EQU      $A020             send task message
    <0000A03C>      XSTP     EQU      $A03C             set/read task priority
    <0000A01C>      XSUI     EQU      $A01C             suspend until event intr
    <0000A02C>      XSUP     EQU      $A02C             enter supervisor mode
    <0000A000>      XSWP     EQU      $A000             swap to next task
    <0000A0B6>      XSZF     EQU      $A0B6             get disk size
    <0000A090>      XTAB     EQU      $A090             tab to column
    <0000A01A>      XTEF     EQU      $A01A             test event flag
    <0000A110>      XTLP     EQU      $A110             xlate logical to phys event
    <0000A036>      XUAD     EQU      $A036             unpack ascii date
    <0000A060>      XUDT     EQU      $A060             unpack date
    <0000A0EE>      XULF     EQU      $A0EE             unlock file
    <0000A016>      XULT     EQU      $A016             unlock task
    <0000A008>      XUSP     EQU      $A008             return to user mode
    <0000A062>      XUTM     EQU      $A062             unpack time
    <0000A116>      XVEC     EQU      $A116             set/read exception vector
    <0000A0F0>      XWBF     EQU      $A0F0             write bytes to file
    <0000A064>      XWDT     EQU      $A064             write date
    <0000A0F2>      XWFA     EQU      $A0F2             write file attributes
    <0000A0FC>      XWFP     EQU      $A0FC             write file parameters
    <0000A0F4>      XWLF     EQU      $A0F4             write line to file
```

```
<0000A0C6>      XWSE     EQU      $A0C6              write sector
<0000A066>      XWTM     EQU      $A066              write time
<0000A0F6>      XZFL     EQU      $A0F6              zero file (wipe)
                *
                *******[ TERMINAL SETUP PARAMETERS ]*****************************
                *
                *   Vmeprom TCB terminal setups (see ST cmd in vmeprom)
                *
<C8CA9B59>      V52      EQU      $C8CA9B59          DEC V52
<1A009B3D>      ADM3A    EQU      $1A009B3D          LEER SEIGLER ADM3A
<00000000>      VT100    EQU      $00000000          DEC VT100
<1E1A9B3D>      TVI950   EQU      $1E1A9B3D          TELE VIDIO TVI 950
                *
                *******[ FORCE CPU DEVICE ADDRESS EQUATES ]**********************
                *
                *   PI/T 2 Timer address equates
                *
<FF800E00>      PIT2     EQU  $FF800E00      Force cpu pit2 base address
                *
<FF800E10>      TCR      EQU  PIT2+$10       timer control register
<FF800E1A>      TSR      EQU  PIT2+$1A       timer status register
<FF800E12>      CPR      EQU  PIT2+$12       counter preload register
<FF800E16>      CTR      EQU  PIT2+$16       counter register
                *
                *******[ TEST MACRO DEFINITIONS ]********************************
                *
                PUSH     MACRO
                         MOVEM.L  \1,-(A7)           Push register(s) on stack
                         ENDM
                POP      MACRO
                         MOVEM.L  (A7)+,\1           pop register(s) off stack
                         ENDM
                CLC      MACRO
                         ANDI.B   #$FE,CCR           clear carry flag in CCR
                         ENDM
                STC      MACRO
                         ORI.B    #$1,CCR            set carry flag in CCR
                         ENDM
                SYSCALL  MACRO
                         DC.W     \1                 Vmeprom line a opcode emulation
                         ENDM
                DISPLAY  MACRO
                         LEA      \1,A1              point to start of message
                         SYSCALL  XPLC               put line to console
                         ENDM
                PROMPT   MACRO
                         LEA      \1,A1              point to prompt msg
                         SYSCALL  XPLC               put line to console
                         SYSCALL  XGCR               get char from console
                         SYSCALL  XPCR               put char to console
                         ENDM
                TERM     MACRO
                         MOVE.L   \1,($43A,A6)       set terminal parms in TCB
                         ENDM
                WAIT     MACRO
```

```
                                MOVE.L    \1,D0                load do with delay value
                                BSR       DELAY                and call pit2 delay subr
                                ENDM
                                INCLUDE   DR11WA.EQ
                       *        Base pointer for VME short I/O

      <FBFF0000>       VMEIO    EQU       $FBFF0000            Short I/O on Force Cpu's

                       *        Base Address for Vnet board registers

      <FBFFE000>       DR11     EQU       VMEIO+$E000          Base address of DR11W

                       *        Offset Address Equates for the Dmai module

      <FBFFE000>       CHSR     EQU       DR11+$00             Channel Status Register
      <FBFFE001>       CHER     EQU       DR11+$01             Channel Error Register
      <FBFFE004>       DCR      EQU       DR11+$04             Device Control Register
      <FBFFE005>       OCR      EQU       DR11+$05             Operation Ctrl Register
      <FBFFE007>       CHCR     EQU       DR11+$07             Channel Ctrl Register
      <FBFFE00A>       MTC      EQU       DR11+$0A             Mem. Transfer Count
      <FBFFE00C>       MACH     EQU       DR11+$0C             Mem. Address Count High
      <FBFFE00E>       MACL     EQU       DR11+$0E             Mem. Address Count Low
      <FBFFE025>       DIVR     EQU       DR11+$25             Done Intr Vector Register

                       *        Offset Address Equates for the Bim module

      <FBFFE043>       BICR     EQU       DR11+$43             Berr Intr Ctrl Register
      <FBFFE045>       AICR     EQU       DR11+$45             Attention Intr Ctrl Reg
      <FBFFE047>       DICR     EQU       DR11+$47             Dma Done Intr Ctrl Reg
      <FBFFE04B>       BIVR     EQU       DR11+$4B             Berr Intr Vect Register
      <FBFFE04D>       AIVR     EQU       DR11+$4D             Attention Intr Vect Reg

                       *        Offset Address Equates for Vnet board registers

      <FBFFE060>       BSR      EQU       DR11+$60             Board Status Register
      <FBFFE061>       BCR      EQU       DR11+$61             Board Control Register
      <FBFFE062>       IODR     EQU       DR11+$62             I/O Data Register
      <FBFFE064>       EMA      EQU       DR11+$64             Extended Mem Address Reg      <=
      <FBFFE065>       AMR      EQU       DR11+$65             Address Modifier Reg
      <FBFFE066>       MCR      EQU       DR11+$66             Master Control Register       <=
      <FBFFE069>       BSRA     EQU       DR11+$69             Board Status Reg A (dr11wa)   <=
      <FBFFE06A>       LIDR     EQU       DR11+$6A             Load Input Data Register      <=
      <FBFFE070>       ODP      EQU       DR11+$70             Output Data Register          <=
      <FBFFE078>       IDP      EQU       DR11+$78             Input  Data Register          <=

                       *        DR11WA command equates

      <00000000>       NULL     EQU       $0000                null (sync) cmd
      <00000001>       ACKN     EQU       $0001                acknowledge cmd
      <00000002>       LREQ     EQU       $0002                link request cmd
      <00000003>       BYTE     EQU       $0003                set byte dma mode
      <00000004>       WORD     EQU       $0004                set word dma mode
      <00000005>       LONG     EQU       $0005                set long dma mode
      <00000006>       SEND     EQU       $0006                Go dma Transmit
      <00000007>       RCVE     EQU       $0007                Go dma receive
```

```
       <00000008>       BRST     EQU       $0008               set burst dma
       <00000009>       CYCL     EQU       $0009               set cycle dma
       <0000000A>       EXIT     EQU       $000A               set exit loop flag
       <0000000B>       ABRT     EQU       $000B               set abort mode flag
       <0000000C>       SWAP     EQU       $000C               set swap command flag
       <0000000D>       BADC     EQU       $000D               bad cmd ack to master

                        **********************************************************************
                        *
                        *        Swap to supervisor to get VBR then back ...

/00
000000                  DR11A4   SYSCALL   XSUP                Supervisor mode on 29
000002 4E7A8801                  MOVEC     VBR,A0              get Vector Base pointer
000006                           SYSCALL   XUSP                back to User mode

                        *        Init Interrupt Vector Table   (@ VBR)

000008 217C00000000+    INITS    MOVE.L    #DONE,($150,A0)     Init Done Vector Addr ($54)
000010 217C00000000+             MOVE.L    #DERR,($154,A0)     Init Derr Vector Addr ($55)
000018 217C00000000+             MOVE.L    #BERR,($158,A0)     Init Berr Vector Addr ($56)
000020 217C00000000+             MOVE.L    #ATTN,($15C,A0)     Init Attn Vector Addr ($57)

                        *        Init Interrupt vector registers

000028 13FC0054+                 MOVE.B    #$54,DIVR           Init Done Intr vector reg
000030 13FC0056+                 MOVE.B    #$56,BIVR           Init Berr Intr vector reg

                        *        Init program variables

000038 23FC00000000+             MOVE.L    #DTBL,DPTR          init test data pointer
000042 42B900000000              CLR.L     PCOUNT              clear passed count
000048 42B900000000              CLR.L     FCOUNT              clear failed count

                        **********************************************************************
                        *                                                                    *
                        *       SHOW WE'RE HERE AND PROMPT FOR START AS MASTER OR SLAVE      *
                        *                                                                    *
                        **********************************************************************

00004E                  DR11A    DISPLAY   DR11_M              show dr11 dma testing
000056                  SELECT   PROMPT    SLCT_M              display select mode msg
000062 0C000058                  CMP.B     #'X',D0             select 'X' for exit this ..
000066 6718                      BEQ.S     DR11W_X             then exit to main menu
000068 0C00004D                  CMP.B     #'M',D0             start in Master mode?
00006C 67000056                  BEQ       M_MODE              then start as master
000070 0C000053                  CMP.B     #'S',D0             start in Slave  mode?
000074 670C                      BEQ.S     S_MODE              then start as slave
000076                           DISPLAY   ARGH_M              for an invalid selection
00007E 60D6                      BRA.S     SELECT              and repeat selection
000080 4E75             DR11W_X  RTS                           return to main menu

                        **********************************************************************
                        *                                                                    *
                        *        S: START THIS NODE IN SLAVE MODE TRANSFER TESTS             *
```

```
                         *                                                                       *
                         *************************************************************************

000082 13FC002C+         S_MODE   MOVE.B    #$2C,BCR            Fled_off,dir_in,en_attn
00008A 13FC0000+                  MOVE.B    #$00,BICR           disable Berr interrupt
000092 13FC0057+                  MOVE.B    #$57,AIVR           Set Attn intr vector
00009A 13FC001E+                  MOVE.B    #$1E,AICR           enable Attn interrupt
0000A2 61000062          S_LOOP   BSR       SLAVE               slave/responder mode
0000A6 08390007+                  BTST      #7,TEST+0           Master mode enabled ?
0000AE 6704                       BEQ.S     S_ONLY              then do Slave only
0000B0 610000A0                   BSR       MASTER              master/initiator mode
0000B4 08390001+         S_ONLY   BTST      #1,FLAGS            check loop on test flag
0000BC 6704                       BEQ.S     S_EXIT              if not set then exit else
0000BE                            SYSCALL   XGCC                check for key entry to
0000C0 67E0                       BEQ.S     S_LOOP              continue or exit test
0000C2 4E75              S_EXIT   RTS                           return from module

                         *************************************************************************
                         *                                                                       *
                         *        M: START THIS NODE IN MASTER MODE TRANSFER TESTS               *
                         *                                                                       *
                         *************************************************************************

0000C4 13FC002C+         M_MODE   MOVE.B    #$2C,BCR            Fled_off,dir_in,en_attn
0000CC 13FC0000+                  MOVE.B    #$00,BICR           don't enable BERR inter
0000D4 13FC0057+                  MOVE.B    #$57,AIVR           Set Attn intr vector
0000DC 13FC001E+                  MOVE.B    #$1E,AICR           enable ATTN interrupt
0000E4 6100006C          M_LOOP   BSR       MASTER              master/initiator mode
0000E8 08390006+                  BTST      #6,TEST+0           Slave mode enabled ?
0000F0 6704                       BEQ.S     M_ONLY              then do Master only
0000F2 61000012                   BSR       SLAVE               slave/responder mode
0000F6 08390001+         M_ONLY   BTST      #1,FLAGS            check loop on test flag
0000FE 6704                       BEQ.S     M_EXIT              if not set then exit else
000100                            SYSCALL   XGCC                check for key entry to
000102 67E0                       BEQ.S     M_LOOP              continue or exit test
000104 4E75              M_EXIT   RTS                           return from module

                         *************************************************************************
                         *                                                                       *
                         *        SLAVE or RESPONDER Sequence                                    *
                         *                                                                       *
                         *************************************************************************

000106                   SLAVE    DISPLAY   SLAV_M              show we are in slave mode
00010E 6100044E          S0       BSR       GET_C               waiting for command input
000112 02F90000+                  CMP2.W    VALID,D0            for a valid command word
00011A 651C                       BCS.S     S1                  ignore invalid command
00011C 41F900000000               LEA       SCMDS,A0            point to slave cmd table
000122 20700400                   MOVE.L    (A0,D0.W*4),A0      get cmd handler address
000126 4E90                       JSR       (A0)                jump subr to cmd handler
000128 08B90002+                  BCLR      #2,FLAGS            test skip ackn flag because
000130 66DC                       BNE       S0                  some commands shouldn't ackn
000132 61000450                   BSR       SLACK               Send SLave Cmd Acknowledge
000136 60D6                       BRA       S0                  and continue slave mode
```

```
                         *       Handle slave command errors

000138                   S1      DISPLAY   CERR_M              show command error message
000140 323900000000              MOVE.W    COMMAND,D1          get bad command word
000146 61000000                  BSR       DSPWORD             display bad command word
00014A                           SYSCALL   XPCL                put crlf to console
00014C 61000442                  BSR       SLECK               SLave Err acknowledge
000150 60BC                      BRA       S0                  continue slave mode

                         ************************************************************************
                         *                                                                      *
                         *       MASTER or INITIATOR Sequence                                   *
                         *                                                                      *
                         ************************************************************************

000152                   MASTER  DISPLAY   MAST_M              show we are in master mode

00015A                   T_LINK  DISPLAY   LREQ_M              show requesting link
000162 303C0002                  MOVE.W    #LREQ,D0            link request command
000166 610003EC                  BSR       PUT_C               put command to slave
00016A 0C400001                  CMP.W     #ACKN,D0            did we get an acknowledge
00016E 670C                      BEQ.S     T_GRNT              then continue
000170                           DISPLAY   NACK_M              get not_ackn (refused)
000178 60000250                  BRA       T_EXIT              then exit
00017C                   T_GRNT  DISPLAY   LACK_M              show Granted

                         *----------------------------------------------------------------------*
                         *       Test Burst and Cycle in main test loop                         *
                         *----------------------------------------------------------------------*

                         *   I.  Execute all dma tests in BURST mode

000184 08390005+         T_BURST BTST      #5,TEST+0           Burst mode enabled ?
00018C 6716                      BEQ.S     T_CYCLE             then goto next test
00018E 6100057C                  BSR       SETBRST             set burst mode
000192 49F900000000              LEA       BRST_M,A4           burst mode msg
000198 303C0008                  MOVE.W    #BRST,D0            slave burst command
00019C 610003B6                  BSR       PUT_C               put command
0001A0 61000250                  BSR       T_MAIN              test loop

                         *   II. Execute all dma tests in CYCLE mode

0001A4 08390004+         T_CYCLE BTST      #4,TEST+0           Burst mode enabled ?
0001AC 6716                      BEQ.S     T_SWAP              then goto next test
0001AE 6100058E                  BSR       SETCYCL             set cycle mode
0001B2 49F900000000              LEA       CYCL_M,A4           cycle mode msg
0001B8 303C0009                  MOVE.W    #CYCL,D0            slave cycle command
0001BC 61000396                  BSR       PUT_C               put command
0001C0 61000230                  BSR       T_MAIN              test loop

                         *   III. Set Burst & Long for data Swapping Tests

0001C4 61000546          T_SWAP  BSR       SETBRST             set burst mode
0001C8 303C0008                  MOVE.W    #BRST,D0            slave burst command
0001CC 61000386                  BSR       PUT_C               put command
```

```
0001D0 6100067A                    BSR       SETLONG             set up for longs
0001D4 303C0005                    MOVE.W    #LONG,D0            slave long command
0001D8 6100037A                    BSR       PUT_C               put command, get ackn

                      *   1.   Test word swap output mode

0001DC 08390000+      T_WSO        BTST      #0,TEST+0           Swap word out enabled ?
0001E4 6730                        BEQ.S     T_WSI               then goto next test
0001E6 49F900000000                LEA       WSOT_M,A4
0001EC 4BF900000000                LEA       OUTP_M,A5
0001F2 3039FBFFE066                MOVE.W    MCR,D0
0001F8 08C00001                    BSET      #1,D0               set word swap out
0001FC 33C0FBFFE066                MOVE.W    D0,MCR
000202 61000250                    BSR       T_TEST
000206 3039FBFFE066                MOVE.W    MCR,D0
00020C 08800001                    BCLR      #1,D0               clr word swap out
000210 33C0FBFFE066                MOVE.W    D0,MCR

                      *   2.   Test word swap input mode

000216 08390007+      T_WSI        BTST      #7,TEST+1           Swap word in enabled ?
00021E 6730                        BEQ.S     T_BSO               then goto next test
000220 49F900000000                LEA       WSIT_M,A4
000226 4BF900000000                LEA       INPT_M,A5
00022C 3039FBFFE066                MOVE.W    MCR,D0
000232 08C00003                    BSET      #3,D0               set word swap in
000236 33C0FBFFE066                MOVE.W    D0,MCR
00023C 61000216                    BSR       T_TEST
000240 3039FBFFE066                MOVE.W    MCR,D0
000246 08800003                    BCLR      #3,D0               clr word swap in
00024A 33C0FBFFE066                MOVE.W    D0,MCR

                      *   3.   Test byte swap ouput mode

000250 08390006+      T_BSO        BTST      #6,TEST+1           Swap byte out enabled ?
000258 6740                        BEQ.S     T_BSI               then goto next test
00025A 49F900000000                LEA       BSOT_M,A4
000260 4BF900000000                LEA       OUTP_M,A5
000266 303C000C                    MOVE.W    #SWAP,D0            slave toggle cmd swap
00026A 610002E8                    BSR       PUT_C
00026E 3039FBFFE066                MOVE.W    MCR,D0
000274 08C00000                    BSET      #0,D0               set byte swap out
000278 33C0FBFFE066                MOVE.W    D0,MCR
00027E 610001D4                    BSR       T_TEST
000282 303C000C                    MOVE.W    #SWAP,D0            slave toggle cmd swap
000286 610002CC                    BSR       PUT_C
00028A 3039FBFFE066                MOVE.W    MCR,D0
000290 08800000                    BCLR      #0,D0               clr byte swap out
000294 33C0FBFFE066                MOVE.W    D0,MCR

                      *   4.   Test byte swap input mode

00029A 08390005+      T_BSI        BTST      #5,TEST+1           Swap byte in enabled ?
0002A2 6730                        BEQ.S     T_WBO               then goto next test
0002A4 49F900000000                LEA       BSIT_M,A4
```

```
0002AA 4BF900000000                LEA       INPT_M,A5
0002B0 3039FBFFE066                MOVE.W    MCR,D0
0002B6 08C00002                    BSET      #2,D0                 set byte swap in
0002BA 33C0FBFFE066                MOVE.W    D0,MCR
0002C0 61000192                    BSR       T_TEST
0002C4 3039FBFFE066                MOVE.W    MCR,D0                clr byte swap in
0002CA 08800002                    BCLR      #2,D0
0002CE 33C0FBFFE066                MOVE.W    D0,MCR

                         *    5.   Test word/byte swap output mode

0002D4 08390004+         T_WBO     BTST      #4,TEST+1             Swap word/byte out enabled ?
0002DC 6748                        BEQ.S     T_WBI                 then goto next test
0002DE 49F900000000                LEA       WBOT_M,A4
0002E4 4BF900000000                LEA       SWPO_M,A5
0002EA 303C000C                    MOVE.W    #SWAP,D0              slave toggle cmd swap
0002EE 61000264                    BSR       PUT_C
0002F2 3039FBFFE066                MOVE.W    MCR,D0
0002F8 08C00001                    BSET      #1,D0                 set word swap out
0002FC 08C00000                    BSET      #0,D0                 set byte swap out
000300 33C0FBFFE066                MOVE.W    D0,MCR
000306 6100014C                    BSR       T_TEST
00030A 303C000C                    MOVE.W    #SWAP,D0              slave toggle cmd swap
00030E 61000244                    BSR       PUT_C
000312 3039FBFFE066                MOVE.W    MCR,D0
000318 08800001                    BCLR      #1,D0                 clr word swap out
00031C 08800000                    BCLR      #0,D0                 clr byte swap out
000320 33C0FBFFE066                MOVE.W    D0,MCR

                         *    6.   Test word/byte swap input mode

000326 08390003+         T_WBI     BTST      #3,TEST+1             Swap word/byte in enabled ?
00032E 6738                        BEQ.S     T_WBIO                then goto next test
000330 49F900000000                LEA       WBIT_M,A4
000336 4BF900000000                LEA       SWPI_M,A5
00033C 3039FBFFE066                MOVE.W    MCR,D0
000342 08C00003                    BSET      #3,D0                 set word swap in
000346 08C00002                    BSET      #2,D0                 set byte swap in
00034A 33C0FBFFE066                MOVE.W    D0,MCR
000350 61000102                    BSR       T_TEST
000354 3039FBFFE066                MOVE.W    MCR,D0
00035A 08800003                    BCLR      #3,D0                 clr word swap in
00035E 08800002                    BCLR      #2,D0                 clr byte swap in
000362 33C0FBFFE066                MOVE.W    D0,MCR

                         *    7.   Test word/byte swap output/input mode

000368 08390002+         T_WBIO    BTST      #2,TEST+1             Swap word/byte out/in enabled ?
000370 6758                        BEQ.S     T_EXIT                then goto next test
000372 49F900000000                LEA       WBIO_M,A4
000378 4BF900000000                LEA       SWIO_M,A5
00037E 303C000C                    MOVE.W    #SWAP,D0              slave toggle cmd swap
000382 610001D0                    BSR       PUT_C
000386 3039FBFFE066                MOVE.W    MCR,D0
00038C 08C00003                    BSET      #3,D0                 set word swap in
```

```
000390 08C00002               BSET      #2,D0             set byte swap in
000394 08C00001               BSET      #1,D0             set word swap out
000398 08C00000               BSET      #0,D0             set byte swap out
00039C 33C0FBFFE066           MOVE.W    D0,MCR
0003A2 610000B0               BSR       T_TEST
0003A6 303C000C               MOVE.W    #SWAP,D0          slave toggle cmd swap
0003AA 610001A8               BSR       PUT_C
0003AE 3039FBFFE066           MOVE.W    MCR,D0
0003B4 08800003               BCLR      #3,D0             clr word swap in
0003B8 08800002               BCLR      #2,D0             clr byte swap in
0003BC 08800001               BCLR      #1,D0             clr word swap out
0003C0 08800000               BCLR      #0,D0             clr byte swap out
0003C4 33C0FBFFE066           MOVE.W    D0,MCR

                      *   IV. Send EXIT loop command to slave

0003CA                T_EXIT  DISPLAY   RELL_M            show releasing link
0003D2 02390003+              AND.B     #$03,FLAGS        clr flags-(loop,break)
0003DA 08390001+              BTST      #1,FLAGS          loop on test flag
0003E2 6704                   BEQ.S     T_LESS            then skip loop info
0003E4 61000582               BSR       L_INFO            else display loop info
0003E8 303C000A       T_LESS  MOVE.W    #EXIT,D0          slave exit command
0003EC 61000166               BSR       PUT_C             put command wait ackn
0003F0 4E75                   RTS                         return from master mode

                      *-------------------------------------------------------------------*
                      *       Main Test loop (longs words and bytes)                      *
                      *-------------------------------------------------------------------*

       <0003F2/00>    T_MAIN  EQU       *                 main test loop entry point

                      *   A.  Test Long Dma Transfers     (32 bits)

0003F2 08390003+      T_LONG  BTST      #3,TEST+0         Long data enabled ?
0003FA 6716                   BEQ.S     T_WORD            then goto next test
0003FC 6100044E               BSR       SETLONG           set up for longs
000400 4BF900000000           LEA       LONG_M,A5         long message
000406 303C0005               MOVE.W    #LONG,D0          slave long command
00040A 61000148               BSR       PUT_C             put command, get ackn
00040E 61000044               BSR       T_TEST            data test loop

                      *   B.  Test Word Dma Transfers     (16 bits)

000412 08390002+      T_WORD  BTST      #2,TEST+0         Word data enabled ?
00041A 6716                   BEQ.S     T_BYTE            then goto next test
00041C 61000458               BSR       SETWORD           set up for words
000420 4BF900000000           LEA       WORD_M,A5         word message
000426 303C0004               MOVE.W    #WORD,D0          slave word command
00042A 61000128               BSR       PUT_C             put command, get ackn
00042E 61000024               BSR       T_TEST            data test loop

                      *   C.  Test Byte Dma Transfers     (8 bits)

000432 08390001+      T_BYTE  BTST      #1,TEST+0         Byte data enabled ?
00043A 6716                   BEQ.S     X_MAIN            then goto next test
```

```
00043C 61000462                 BSR      SETBYTE             set up for bytes
000440 4BF900000000             LEA      BYTE_M,A5           byte message
000446 303C0003                 MOVE.W   #BYTE,D0            slave byte command
00044A 61000108                 BSR      PUT_C               put command, get ackn
00044E 61000004                 BSR      T_TEST              data test loop

                       *   D.   Then we return for cycle or exit testing

000452 4E75            X_MAIN   RTS                          return to caller

                       *-------------------------------------------------------------------*
                       *        Main Dma I/O output,input compare test data loop           *
                       *-------------------------------------------------------------------*

                       *   1.   Display Test mode (Burst/Cycle/Swap) and (B/W/I/O)

000454 224C            T_TEST   MOVE.L   A4,A1               get mode message
000456                          SYSCALL  XPLC                put line to console
000458 224D                     MOVE.L   A5,A1               get data size or IO message
00045A                          SYSCALL  XPLC                put line to console

                       *   2.   Fill test buffer with TDATA (test data var)

00045C 41F900000000    T_FILL   LEA      FILLBUFF,A0         w/TDATA update data ptr
000462 08390001+                BTST     #1,TEST+1           Data tables enabled
00046A 6606                     BNE.S    T_F0                then use fill buffer
00046C 41F900000000             LEA      FILLMEM,A0          w/o TDATA update data ptr
000472 4E90            T_F0     JSR      (A0)                do fill whatever

                       *   3.   Transmit the buffer to the Slave board

000474 303C0007        T_TX     MOVE.W   #RCVE,D0            slave Rx command
000478 610000DA                 BSR      PUT_C               put cmd wait ackn
00047C 610001B6                 BSR      TRANSMIT            to slave board
000480 08B90007+                BCLR     #7,FLAGS            tx error
000488 66000054                 BNE      D_FAIL              then dma error

                       *   4.   Receive the buffer from the Slave board

00048C 303C0006        T_RX     MOVE.W   #SEND,D0            Slave Tx command
000490 610000C2                 BSR      PUT_C               put cmd wait ackn
000494 610001F8                 BSR      RECEIVE             from slave board
000498 08B90007+                BCLR     #7,FLAGS            test error flag
0004A0 6600003C                 BNE      D_FAIL              then dma error

                       *   5.   Test the buffer for correct TDATA

0004A4 6100053C        T_BUFF   BSR      TESTBUFF            for t_data
0004A8 08B90007+                BCLR     #7,FLAGS            buffer err?
0004B0 6600003E                 BNE      X_FAIL              then fail

                       *   6.   Test for last data in test data table

0004B4 08390001+       T_NEXT   BTST     #1,TEST+1           data tables enabled
0004BC 670C                     BEQ.S    T_PASS              then skip last data test
```

```
0004BE 0CB900000000+            CMPI.L    #0,TDATA            last data ?
0004C8 6692                     BNE       T_FILL              then refill & test

                        *   7.  Show passed and exit

0004CA                  T_PASS  DISPLAY   PASS_M              else show passed
0004D2 06B900000001+            ADDI.L    #1,PCOUNT           increment pass count
0004DC 4E75                     RTS

                        *   8.  Handle dmai error from slave board

0004DE                  D_FAIL  DISPLAY   BADD_M              bad dma from slave
0004E6 6008                     BRA.S     X_FAIL              common exit

                        *   9.  Handle acknowledge error from slave board

0004E8                  A_FAIL  DISPLAY   BADA_M              bad ackn from slave

                        *       Common exit for transfer tests

0004F0 06B900000001+    X_FAIL  ADDI.L    #1,FCOUNT           increment fail count
0004FA                          DISPLAY   CHSR_M              Common exit
000502 1239FBFFE000             MOVE.B    CHSR,D1
000508 61000000                 BSR       DSPBYTE
00050C                          DISPLAY   STAT_M
000514 123900000000             MOVE.B    STATS,D1
00051A 61000000                 BSR       DSPBYTE
00051E                          DISPLAY   FLAG_M
000526 123900000000             MOVE.B    FLAGS,D1
00052C 61000000                 BSR       DSPBYTE
000530 23FC00000000+            MOVE.L    #DTBL,DPTR
00053A                          SYSCALL   XPCL
00053C 08390000+                BTST      #0,FLAGS            set this in MAIN
000544 670C                     BEQ.S     NO_BR               break flag not set
000546                  DO_BR   PROMPT    ANYK_M              break flag set
000552 4E75             NO_BR   RTS                           return errors

                        **********************************************************************
                        ***                                                                ***
                        **                     DR11W Board Subroutines                      **
                        ***                                                                ***
                        **********************************************************************

                        *--------------------------------------------------------------------*
                        *       PUT/GET commands to/from other board (master & slave)        *
                        *--------------------------------------------------------------------*

000554 33C0FBFFE062     PUT_C   MOVE.W    D0,IODR             write commmand via IODR
00055A 6100003C                 BSR       ATTEN               set attn to other board
00055E 08B90006+        GET_C   BCLR      #6,FLAGS            clear and test attn flag
000566 67F6                     BEQ.S     GET_C               wait for attn from other brd
000568 303900000000             MOVE.W    COMMAND,D0          and get command from buffer
00056E 08390004+                BTST      #4,FLAGS            test swap command bytes flag
000576 6702                     BEQ.S     GCX                 skip swapping command bytes
000578 E058                     ROR.W     #8,D0               else swap command bytes
```

```
00057A 13FC001E+        GCX       MOVE.B    #$1E,AICR           re-enable attn interrupt
000582 4E75                       RTS

                        *----------------------------------------------------------------------*
                        *         SLave ACKn to other board with Atten (fall thru)             *
                        *----------------------------------------------------------------------*

000584 33FC0001+        SLACK     MOVE.W    #ACKN,IODR          put acknowledge in IODR
00058C 6000000A                   BRA       ATTEN               else to ackn atten to Master

                        *----------------------------------------------------------------------*
                        *         SLave ERR ACKn to other board with Atten                     *
                        *----------------------------------------------------------------------*

000590 33FC000D+        SLECK     MOVE.W    #BADC,IODR          put bac cmd ackn in IODR

                        *----------------------------------------------------------------------*
                        *         Toggle attn input to other board (both)                      *
                        *----------------------------------------------------------------------*

000598 1039FBFFE061     ATTEN     MOVE.B    BCR,D0              get board control status
00059E 020000FC                   AND.B     #$FC,D0             mask cycle and go bits
0005A2 08C00006                   BSET      #6,D0               set attention bit high
0005A6 13C0FBFFE061               MOVE.B    D0,BCR              write to other board
0005AC 08800006                   BCLR      #6,D0               set attention bit low
0005B0 13C0FBFFE061               MOVE.B    D0,BCR              write to other board
0005B6 4E75                       RTS                           return

                        *----------------------------------------------------------------------*
                        *         NULL command handler (slave)                                 *
                        *----------------------------------------------------------------------*

0005B8                  NULLCMD   DISPLAY   NULL_M              show doing null command
0005C0 4E75                       RTS

                        *----------------------------------------------------------------------*
                        *         LINK request command handler (slave)                         *
                        *----------------------------------------------------------------------*

0005C2                  LINKCMD   DISPLAY   LINK_M              show link request from other
0005CA                            DISPLAY   GRNT_M              show doing grant to other
0005D2 4E75                       RTS

                        *----------------------------------------------------------------------*
                        *         ACKN command handler (slave) (if he should get one)          *
                        *----------------------------------------------------------------------*

0005D4                  ACKNCMD   DISPLAY   ACKN_M              show doing ackn command
0005DC 4E75                       RTS

                        *----------------------------------------------------------------------*
                        *         BADC command handler (slave) (if he should get one)          *
                        *----------------------------------------------------------------------*

0005DE                  BADCMD    DISPLAY   BCMD_M              show doing BADC command
```

```
0005E6 4E75                         RTS

                          *---------------------------------------------------------------------*
                          *       Set Exit current loop flag (slave)                            *
                          *---------------------------------------------------------------------*

0005E8                    EXITCMD   DISPLAY   EXIT_M              show exit message
0005F0 02390003+                    ANDI.B    #$03,FLAGS          clr flags except loop & break
0005F8 618A                         BSR       SLACK               and acknowledge exit
0005FA DFFC00000004                 ADDA.L    #4,A7               return two levels from here
000600 4E75                         RTS                           & return

                          *---------------------------------------------------------------------*
                          *       Abort Program handler (quit) (slave)                          *
                          *---------------------------------------------------------------------*

000602                    ABRTCMD   DISPLAY   ABRT_M              show aborting message
00060A 02390003+                    ANDI.B    #$03,FLAGS          clr flags except loop & break
000612 6100FF70                     BSR       SLACK               and acknowledge exit
000616 DFFC00000008                 ADDA.L    #8,A7               return three levels from here
00061C 4E75                         RTS                           all the way back to main menu!

                          *---------------------------------------------------------------------*
                          *       Swap command bytes (slave)                                    *
                          *---------------------------------------------------------------------*

00061E 08790004+          SWAPCMD   BCHG      #4,FLAGS            toggle swap command flag
000626 4E75                         RTS

                          *---------------------------------------------------------------------*
                          *       Init Dmai and BCR to Transmit data (both)                     *
                          *---------------------------------------------------------------------*
                          *
                          *   slave transmit entry point, fall thru ...
                          *
000628 33FC0001+          SLAVE_TX  MOVE.W    #ACKN,IODR          tell master we got the cmd
000630 6100FF66                     BSR       ATTEN               via attention interrupt
                          *
                          *   master transmit entry point
                          *
000634 08F90002+          TRANSMIT  BSET      #2,FLAGS            set skip cmd ackn for slave
00063C 1039FBFFE061                 MOVE.B    BCR,D0              Get BCR status
000642 020000FC                     AND.B     #$FC,D0             mask cycle and go bits
000646 08800005                     BCLR      #5,D0               set direction to out
00064A 13C0FBFFE061                 MOVE.B    D0,BCR              set direction in BCR
000650 08800003                     BCLR      #3,D0               set FLED on for effect
000654 08C00000                     BSET      #0,D0               set go bit in BCR image
000658 08C00001                     BSET      #1,D0               set cycle bit in image
00065C 207900000000                 MOVE.L    TXS,A0              get pointer to setup
000662 6100006A                     BSR       SETDMA              do dmai module setup
000666 08390004+          TX_RDY    BTST      #4,BSRA             read board status reg
00066E 67F6                         BEQ.S     TX_RDY              if not ready then wait
000670 13C0FBFFE061                 MOVE.B    D0,BCR              set go and cycle to DMA
000676 08B90005+          TX_DONE   BCLR      #5,FLAGS            test done in flags
00067E 67F6                         BEQ.S     TX_DONE             wait for done flag
```

```
000680 4E75                      RTS                            return to caller

                         *-----------------------------------------------------------------*
                         *        Init Dmai and BCR to Receive data (both)                  *
                         *-----------------------------------------------------------------*
                         *
                         *   slave receive entry point, fall thru ...
                         *
000682 33FC0001+         SLAVE_RX MOVE.W    #ACKN,IODR          tell master we got the command
00068A 6100FF0C                   BSR       ATTEN               via attention interrupt
                         *
                         *   master receive entry point
                         *
00068E 08F90002+         RECEIVE  BSET      #2,FLAGS            set skip cmd ackn for slave
000696 1039FBFFE061               MOVE.B    BCR,D0              Get BCR status
00069C 020000FC                   AND.B     #$FC,D0             mask cycle and go bits
0006A0 08C00005                   BSET      #5,D0               set direction to in
0006A4 13C0FBFFE061               MOVE.B    D0,BCR              set direction in BCR
0006AA 08800003                   BCLR      #3,D0               set FLED on for effect
0006AE 08C00000                   BSET      #0,D0               set go bit in BCR image
0006B2 207900000000               MOVE.L    RXS,A0              point to Rx setup table
0006B8 61000014                   BSR       SETDMA              and go set the dmai
0006BC 13C0FBFFE061               MOVE.B    D0,BCR              set go in BCR to DMA
0006C2 08B90005+         RX_DONE  BCLR      #5,FLAGS            test done in flags
0006CA 67F6                       BEQ.S     RX_DONE             wait for done flag
0006CC 4E75                       RTS                           return to caller

                         *-----------------------------------------------------------------*
                         *        Setup Dmai for transfers from table @ (A0) (both)         *
                         *-----------------------------------------------------------------*
                         * registers:  csr|dcr|ocr|amr|ema|ccr| mtc.w | mac.h | mac.l |dicr
                         *               0 | 1 | 2 | 3 | 4 | 5 | 6   7 | 8   9 | A   B | C

0006CE 13D8FBFFE000      SETDMA   MOVE.B    (A0)+,CHSR          Channel Status Register
0006D4 13D8FBFFE004               MOVE.B    (A0)+,DCR           Device Control Registr
0006DA 13D8FBFFE005               MOVE.B    (A0)+,OCR           Operation Control Reg
0006E0 13D8FBFFE065               MOVE.B    (A0)+,AMR           Address Modifier reg
0006E6 13D8FBFFE064               MOVE.B    (A0)+,EMA           Extended Memory Address
0006EC 13D8FBFFE007               MOVE.B    (A0)+,CHCR          Enable dma starts & intrs
0006F2 33D8FBFFE00A               MOVE.W    (A0)+,MTC           Mem Transfer Count
0006F8 33D8FBFFE00C               MOVE.W    (A0)+,MACH          Mem Address Count (hi)
0006FE 33D8FBFFE00E               MOVE.W    (A0)+,MACL          Mem Address Count (lo)
000704 13D8FBFFE047               MOVE.B    (A0)+,DICR          Done Intr Control Reg
00070A 4E75                       RTS                           return to caller

                         *-----------------------------------------------------------------*
                         *        Set BURST mode in Transfers tables (both)                 *
                         *-----------------------------------------------------------------*

00070C 08B90007+         SETBRST  BCLR      #7,TX.L+1           set Burst mode in tables (dcr)
000714 08B90007+                  BCLR      #7,RX.L+1
00071C 08B90007+                  BCLR      #7,TX.W+1
000724 08B90007+                  BCLR      #7,RX.W+1
00072C 08B90007+                  BCLR      #7,TX.B+1
000734 08B90007+                  BCLR      #7,RX.B+1
```

```
00073C 4E75                      RTS

                         *-------------------------------------------------------------------*
                         *       Set CYCLE mode in Transfer tables (both)                    *
                         *-------------------------------------------------------------------*

00073E 08F90007+         SETCYCL BSSET   #7,TX.L+1           set Cycle mode in tables (dcr)
000746 08F90007+                 BSET    #7,RX.L+1
00074E 08F90007+                 BSET    #7,TX.W+1
000756 08F90007+                 BSET    #7,RX.W+1
00075E 08F90007+                 BSET    #7,TX.B+1
000766 08F90007+                 BSET    #7,RX.B+1
00076E 4E75                      RTS

                         *-------------------------------------------------------------------*
                         *       Set Address Modifier in Transfer tables (both)              *
                         *-------------------------------------------------------------------*

000770 13C100000000      SETAMR  MOVE.B  D1,TX.L+3           set AMR in tables for long
000776 13C100000000              MOVE.B  D1,RX.L+3
00077C 08C10006                  BSET    #6,D1               set not longword bit in AMR &
000780 13C100000000              MOVE.B  D1,TX.W+3           set AMR in tables for word/byte
000786 13C100000000              MOVE.B  D1,RX.W+3
00078C 13C100000000              MOVE.B  D1,TX.B+3
000792 13C100000000              MOVE.B  D1,RX.B+3
000798 4E75                      RTS

                         *-------------------------------------------------------------------*
                         *       Set Extended Memory Address in Transfer tables (both)       *
                         *-------------------------------------------------------------------*

00079A 13C100000000      SETEMA  MOVE.B  D1,TX.L+4           set EMA in tables
0007A0 13C100000000              MOVE.B  D1,RX.L+4
0007A6 13C100000000              MOVE.B  D1,TX.W+4
0007AC 13C100000000              MOVE.B  D1,RX.W+4
0007B2 13C100000000              MOVE.B  D1,TX.B+4
0007B8 13C100000000              MOVE.B  D1,RX.B+4
0007BE 4E75                      RTS

                         *-------------------------------------------------------------------*
                         *       Set Memory Transfer Count in tables (MTC.W) (both)          *
                         *-------------------------------------------------------------------*

0007C0 33C100000000      SETMTC  MOVE.W  D1,TX.L+6           set MTC in tables
0007C6 33C100000000              MOVE.W  D1,RX.L+6
0007CC 33C100000000              MOVE.W  D1,TX.W+6
0007D2 33C100000000              MOVE.W  D1,RX.W+6
0007D8 33C100000000              MOVE.W  D1,TX.B+6
0007DE 33C100000000              MOVE.W  D1,RX.B+6
0007E4 4E75                      RTS

                         *-------------------------------------------------------------------*
                         *       Set Memory Address Counter in Transfer tables (both)        *
                         *-------------------------------------------------------------------*
```

```
0007E6 23C100000000     SETMAC  MOVE.L   D1,TX.L+8           set Mac in tables
0007EC 23C100000000             MOVE.L   D1,RX.L+8
0007F2 23C100000000             MOVE.L   D1,TX.W+8
0007F8 23C100000000             MOVE.L   D1,RX.W+8
0007FE 23C100000000             MOVE.L   D1,TX.B+8
000804 23C100000000             MOVE.L   D1,RX.B+8
00080A 4E75                     RTS

                        *--------------------------------------------------------------------*
                        *       Set Transfer defaults for MTC, MAC, AMR, EMA (menu)          *
                        *--------------------------------------------------------------------*

00080C 223CFB000000     SETRR2  MOVE.L   #$FB000000,D1       default transfer buffer
000812 61D2                     BSR      SETMAC              set MAC
000814 323CFFFF                 MOVE.W   #$FFFF,D1           default transfer count
000818 61A6                     BSR      SETMTC              set MTC
00081A 123C0039                 MOVE.B   #$39,D1             default address modifier
00081E 6100FF50                 BSR      SETAMR              set AMR
000822 123C00FB                 MOVE.B   #$FB,D1             default extended address
000826 6100FF72                 BSR      SETEMA              set EMA
00082A 4E75                     RTS                          return

00082C 223C000B0000     SETC33  MOVE.L   #$000B0000,D1       default transfer buffer
000832 61B2                     BSR      SETMAC              set MAC
000834 323CFFFF                 MOVE.W   #$FFFF,D1           default transfer count
000838 6186                     BSR      SETMTC              set MTC
00083A 123C0009                 MOVE.B   #$09,D1             default address modifier
00083E 6100FF30                 BSR      SETAMR              set AMR
000842 123C00FF                 MOVE.B   #$FF,D1             default extended address
000846 6100FF52                 BSR      SETEMA              set EMA
00084A 4E75                     RTS                          return

                        *--------------------------------------------------------------------*
                        *       Set Transfer mode pointers to LONG mode (both)               *
                        *--------------------------------------------------------------------*

00084C 23FC00000000+    SETLONG MOVE.L   #FILL_L,FBS         setup fill buffer subr
000856 23FC00000000+            MOVE.L   #TEST_L,TBS         setup test buffer subr
000860 23FC00000000+            MOVE.L   #TX.L,TXS           setup word xfer table
00086A 23FC00000000+            MOVE.L   #RX.L,RXS           setup word xfer table
000874 4E75                     RTS                          and return

                        *--------------------------------------------------------------------*
                        *       Set Transfer mode pointers to WORD mode (both)               *
                        *--------------------------------------------------------------------*

000876 23FC00000000+    SETWORD MOVE.L   #FILL_W,FBS         setup fill buffer subr
000880 23FC00000000+            MOVE.L   #TEST_W,TBS         setup test buffer subr
00088A 23FC00000000+            MOVE.L   #TX.W,TXS           setup word xfer table
000894 23FC00000000+            MOVE.L   #RX.W,RXS           setup word xfer table
00089E 4E75                     RTS                          and return

                        *--------------------------------------------------------------------*
                        *       Set Transfer mode pointers to BYTE mode (both)               *
                        *--------------------------------------------------------------------*
```

```
0008A0 23FC00000000+   SETBYTE  MOVE.L   #FILL_B,FBS         setup fill buffer subr
0008AA 23FC00000000+            MOVE.L   #TEST_B,TBS         setup test buffer subr
0008B4 23FC00000000+            MOVE.L   #TX.B,TXS           setup byte xfer table
0008BE 23FC00000000+            MOVE.L   #RX.B,RXS           setup byte xfer table
0008C8 4E75                     RTS                          and return

                       *-------------------------------------------------------------------*
                       *        Dump transfer tables to console                            *
                       *-------------------------------------------------------------------*
                       *   (*) Note To the unwary:  This routine is not straight forward.

0008CA                 DUMP_T   DISPLAY  DT0                 show table breakout
0008D2                          DISPLAY  DT1                 show tx long
0008DA 41F900000000             LEA      TX.L,A0
0008E0 303C000C                 MOVE.W   #$C,D0
0008E4 61000076                 BSR      ST02
0008E8                          DISPLAY  DT2                 show tx word
0008F0 41F900000000             LEA      TX.W,A0
0008F6 303C000C                 MOVE.W   #$C,D0
0008FA 61000060                 BSR      ST02
0008FE                          DISPLAY  DT3
000906 41F900000000             LEA      TX.B,A0             show tx byte
00090C 303C000C                 MOVE.W   #$C,D0
000910 6100004A                 BSR      ST02
000914                          DISPLAY  DT4                 show rx long
00091C 41F900000000             LEA      TX.L,A0
000922 303C000C                 MOVE.W   #$C,D0
000926 61000034                 BSR      ST02
00092A                          DISPLAY  DT5                 show rx word
000932 41F900000000             LEA      TX.W,A0
000938 303C000C                 MOVE.W   #$C,D0
00093C 6100001E                 BSR      ST02
000940                          DISPLAY  DT6                 show rx byte
000948 41F900000000             LEA      TX.B,A0
00094E 303C000C                 MOVE.W   #$C,D0
000952 6008                     BRA.S    ST02                fall thru to rts main (*)
000954                 ST01     DISPLAY  DTL                 Delimiter for loop
00095C 1218            ST02     MOVE.B   (A0)+,D1            get table byte
00095E 61000000                 BSR      DSPBYTE             show it
000962 51C8FFF0                 DBF      D0,ST01             do loop count
000966 4E75                     RTS                          return to caller

                       *-------------------------------------------------------------------*
                       *        Display loop on test information                           *
                       *-------------------------------------------------------------------*

000968                 L_INFO   DISPLAY  PCNT_M              show pass message
000970 223900000000             MOVE.L   PCOUNT,D1           get passed count and
000976 61000000                 BSR      DSPLONG             show it as hex longword
00097A                          DISPLAY  FCNT_M              show fail message
000982 223900000000             MOVE.L   FCOUNT,D1           get failed count and
000988 61000000                 BSR      DSPLONG             show it as hex longword
00098C 4E75                     RTS                          and return
```

```
                        *-------------------------------------------------------------------*
                        *       Fill Buffer with Test DATA & incr pointer (master)          *
                        *-------------------------------------------------------------------*
                        *       Fill with TDATA update

00098E 207900000000     FILLBUFF MOVE.L    DPTR,A0                get test data pointer
000994 23D800000000              MOVE.L    (A0)+,TDATA            get test data from table
00099A 6606                      BNE.S     F0                     if data <> 0 then set pointer
00099C 41F900000000              LEA       DTBL,A0                else point to start of table
0009A2 23C800000000     F0       MOVE.L    A0,DPTR                and set test data pointer

                        *       Fill only (no tdata update)

0009A8 207900000000     FILLMEM  MOVE.L    TX.L+8,A0              point to transfer buffer
0009AE 323900000000              MOVE.W    TX.L+6,D1              Get buffer length in D1
0009B4 5341                      SUB.W     #1,D1                  Less 1 for DBF instruction
0009B6 203900000000              MOVE.L    TDATA,D0               get current test data
0009BC 227900000000              MOVE.L    FBS,A1                 pointer to fill block subr
0009C2 4ED1                      JMP       (A1)                   use B/W/L subr & return

                        *       Byte buffer fill subroutine

0009C4 10C0             FILL_B   MOVE.B    D0,(A0)+               put data into buffer
0009C6 4600                      NOT.B     D0                     invert every other data
0009C8 51C9FFFA                  DBF       D1,FILL_B              do the whole buffer
0009CC 4E75                      RTS                              return to caller

                        *       Word Buffer fill subroutine

0009CE 30C0             FILL_W   MOVE.W    D0,(A0)+               put data into buffer
0009D0 4640                      NOT.W     D0                     invert every other data
0009D2 51C9FFFA                  DBF       D1,FILL_W              do the whole buffer
0009D6 4E75                      RTS                              return to caller

                        *       Long Buffer fill subroutine

0009D8 20C0             FILL_L   MOVE.L    D0,(A0)+               put data into buffer
0009DA 4680                      NOT.L     D0                     invert every other data
0009DC 51C9FFFA                  DBF       D1,FILL_L              do the whole buffer
0009E0 4E75                      RTS                              return to caller

                        *-------------------------------------------------------------------*
                        *       Test buffer Data to current test data: TDATA  (master)      *
                        *-------------------------------------------------------------------*

0009E2 207900000000     TESTBUFF MOVE.L    TX.L+8,A0              point to transfer buffer
0009E8 323900000000              MOVE.W    TX.L+6,D1              get buffer length in D1
0009EE 5341                      SUB.W     #1,D1                  Less 1 for DBF instruction
0009F0 203900000000              MOVE.L    TDATA,D0               get current test data and

                        *       handle word and byte swapping capability

0009F6 3439FBFFE066     T0       MOVE.W    MCR,D2                 read master control register
0009FC 08020003                  BTST      #3,D2                  input word swapping set
000A00 6702                      BEQ.S     T1                     then skip it
```

```
000A02 4840                        SWAP      D0                 else swap words in tdata
000A04 08020002        T1          BTST      #2,D2              input byte swapping set
000A08 6708                        BEQ.S     T2                 then skip it
000A0A E058                        ROR.W     #8,D0              else swap bytes 0,1
000A0C 4840                        SWAP      D0                 swap words
000A0E E058                        ROR.W     #8,D0              then swap bytes 2,3
000A10 4840                        SWAP      D0                 restore words
000A12 08020001        T2          BTST      #1,D2              output word swapping set
000A16 6702                        BEQ.S     T3                 then skip it
000A18 4840                        SWAP      D0                 else swap words in tdata
000A1A 08020000        T3          BTST      #0,D2              output byte swapping set
000A1E 6708                        BEQ.S     T4                 then skip it
000A20 E058                        ROR.W     #8,D0              else swap bytes 0,1
000A22 4840                        SWAP      D0                 swap words
000A24 E058                        ROR.W     #8,D0              then swap bytes 2,3
000A26 4840                        SWAP      D0                 restore words

                       *           now get test by size pointer

000A28 227900000000    T4          MOVE.L    TBS,A1             pointer to test buffer subr
000A2E 4ED1                        JMP       (A1)               use B/W/L subr & return

                       *           Byte Buffer Test subroutine

000A30 B018            TEST_B      CMP.B     (A0)+,D0           comp with data in buffer
000A32 6608                        BNE.S     BBX                Error if not equal
000A34 4600                        NOT.B     D0                 invert for testing
000A36 51C9FFF8                    DBF       D1,TEST_B          test the whole buffer
000A3A 4E75                        RTS                          return passing
000A3C 91FC00000001    BBX         SUBA.L    #1,A0              calc error address
000A42 61000076                    BSR       BUFF_E             common buff err sub
000A46 1210                        MOVE.B    (A0),D1            get error byte
000A48 61000000                    BSR       DSPBYTE            show it
000A4C                             DISPLAY   WROT_M             show write msg
000A54 1200                        MOVE.B    D0,D1              get wrote byte
000A56 61000000                    BSR       DSPBYTE            show it
000A5A                             SYSCALL   XPCL               new line
000A5C 4E75                        RTS

                       *           Word Buffer Test subroutine

000A5E B058            TEST_W      CMP.W     (A0)+,D0           comp with data in buffer
000A60 6608                        BNE.S     BWX                Error if not equal
000A62 4640                        NOT.W     D0                 invert for testing
000A64 51C9FFF8                    DBF       D1,TEST_W          test the whole buffer
000A68 4E75                        RTS                          return passing
000A6A 91FC00000002    BWX         SUBA.L    #2,A0              calc error addrsess
000A70 61000048                    BSR       BUFF_E             common buff err sub
000A74 3210                        MOVE.W    (A0),D1            get error word
000A76 61000000                    BSR       DSPWORD            show it
000A7A                             DISPLAY   WROT_M             show wrote msg
000A82 3200                        MOVE.W    D0,D1              get wrote word
000A84 61000000                    BSR       DSPWORD            show it
000A88                             SYSCALL   XPCL               new line
000A8A 4E75                        RTS
```

```
                        *         Long Buffer Test subroutine

000A8C B098             TEST_L  CMP.L     (A0)+,D0               comp with data in buffer
000A8E 6608                     BNE.S     BLX                    Error if not equal
000A90 4680                     NOT.L     D0                     invert for testing
000A92 51C9FFF8                 DBF       D1,TEST_L              test the whole buffer
000A96 4E75                     RTS                              return passing
000A98 91FC00000004     BLX     SUBA.L    #4,A0                  calc error addrsess
000A9E 6100001A                 BSR       BUFF_E                 common buff err sub
000AA2 2210                     MOVE.L    (A0),D1                get error long
000AA4 61000000                 BSR       DSPLONG                show it
000AA8                          DISPLAY   WROT_M                 show wrote msg
000AB0 2200                     MOVE.L    D0,D1                  get wrote long
000AB2 61000000                 BSR       DSPLONG                show it
000AB6                          SYSCALL   XPCL                   new line
000AB8 4E75                     RTS

                        *         Test buffer subroutine error subroutine (sub sub)

000ABA                  BUFF_E  DISPLAY   BADB_M                 bad data from slave
000AC2 08F90007+                BSET      #7,FLAGS               set error bit in flags
000ACA 2208                     MOVE.L    A0,D1                  get fixed addr
000ACC 61000000                 BSR       DSPLONG                show it
000AD0                          DISPLAY   READ_M                 show read data msg
000AD8 4E75                     RTS                              and return

                        *********************************************************************
                        ***                                                               ***
                        **                     DR11WA Interrupt Handlers                   **
                        ***                                                               ***
                        *********************************************************************

                        *-------------------------------------------------------------------*
                        *         Attention Handler (Command input interrupt) (both)        *
                        *-------------------------------------------------------------------*

000ADA 08F90006+        ATTN    BSET      #6,FLAGS               set attn flag for cmd ready
000AE2 33F9FBFFE062+            MOVE.W    IODR,COMMAND           read command via IODR
000AEC 4E73                     RTE                              return from exception

                        *-------------------------------------------------------------------*
                        *         Dma DONE & DERR handler (both)                            *
                        *-------------------------------------------------------------------*

000AEE 08F90007+        DERR    BSET      #7,FLAGS               Set error bit in flags
000AF6 13FC0054+                MOVE.B    #$54,DIVR              restore vector ls_bit
000AFE 08F90005+        DONE    BSET      #5,FLAGS               Set done bit in flags
000B06 13FC002C+                MOVE.B    #$2C,BCR               clean up the BCR
000B0E 13F9FBFFE060+            MOVE.B    BSR,STATS              read bsr into stats
000B18 13F900000000+            MOVE.B    TX.W+3,AMR             AMR must always be not lword
000B22 4E73                     RTE                              Return from Exception

                        *-------------------------------------------------------------------*
                        *         BUS Error (board) Handler (both)                          *
```

```
                         *-----------------------------------------------------------------*

000B24 08F90007+         BERR      BSET      #7,FLAGS           set error bit in flags
000B2C 4E73                        RTE                          and return from exception

                         *******************************************************************
                         ***                                                             ***
                         **                     DR11WA Test messages                      **
                         ***                                                             ***
                         *******************************************************************

000B2E                   DR11_M
000B2E 0D0A0A             DC.B CR,LF,LF
000B31 2A2A2A2A2A2A2A+    DC.B '************************************************************',CR,LF
000B6F 2A202020202020+    DC.B '*                                                          *',CR,LF
000BAD 2A202020202020+    DC.B '*       (((((( VMIVME DR11WA TRANSFER TESTS ))))))         *',CR,LF
000BEB 2A202020202020+    DC.B '*                                                          *',CR,LF
000C29 2A2A2A2A2A2A2A+    DC.B '************************************************************',CR,LF,NUL
000C68                   SLCT_M
000C68 53656C65637420+    DC.B 'Select Transfer Test Startup mode M=Master, S=Slave (M/S): ',NUL
000CA4 203F0D0700        ARGH_M DC.B ' ?',CR,BEL,NUL
000CA9 0D0A              MMOD_M DC.B CR,LF
000CAB 4D415354455220+    DC.B 'MASTER selected. Enter X to abort, any other key to start: ',NUL
000CE7 0D0A              SMOD_M DC.B CR,LF
000CE9 534C4156452020+    DC.B 'SLAVE  Selected. Enter X to abort, any other key to start: ',NUL

000D25 0D0A5072657373+   ANYK_M DC.B CR,LF,'Press any key to continue: ',nul

000D43 0D0A0A203C204D+   MAST_M DC.B CR,LF,LF,' < MASTER MODE >',CR,LF,NUL
000D59 52657175657374+   LREQ_M DC.B 'Requesting Link Mastership ..... ',NUL
000D7B 52656675736564+   NACK_M DC.B 'Refused',CR,LF,NUL
000D85 4772616E746564+   LACK_M DC.B 'Granted',CR,LF,NUL
000D8F 52656C65617369+   RELL_M DC.B 'Releasing Link Mastership to other board',CR,LF,NUL

000DBA 0D0A0A203C2053+   SLAV_M DC.B CR,LF,LF,' < SLAVE  MODE >',CR,LF,NUL
000DD0 52656365697665+   LINK_M DC.B 'Received Link Request from other board',CR,LF,NUL
000DF9 4772616E74696E+   GRNT_M DC.B 'Granting Link Mastership to other board',CR,LF,NUL
000E23 52656365697665+   EXIT_M DC.B 'Received Exit command from link Master',CR,LF,NUL
000E4C 52656365697665+   BCMD_M DC.B 'Received BADC command from link Master',CR,LF,NUL
000E75 52656365697665+   ABRT_M DC.B 'Received Abort command from link Master',CR,LF,NUL
000E9F 52656365697665+   CERR_M DC.B 'Received Bad command from Master = ',nul

000EC3 54657374696E67+   BRST_M DC.B 'Testing Burst Dma ',NUL
000ED6 54657374696E67+   CYCL_M DC.B 'Testing Cycle Dma ',NUL

000EE9 4C6F6E67206461+   LONG_M DC.B 'Long data ..... ',nul
000EFA 576F7264206461+   WORD_M DC.B 'Word data ..... ',nul
000F0B 42797465206461+   BYTE_M DC.B 'Byte data ..... ',nul

000F1C 54657374696E67+   WSOT_M DC.B 'Testing word swap ',nul
000F2F 54657374696E67+   WSIT_M DC.B 'Testing word swap ',nul
000F42 54657374696E67+   BSOT_M DC.B 'Testing byte swap ',nul
000F55 54657374696E67+   BSIT_M DC.B 'Testing byte swap ',nul
000F68 54657374696E67+   WBOT_M DC.B 'Testing word/byte ',nul
000F7B 54657374696E67+   WBIT_M DC.B 'Testing word/byte ',nul
```

```
000F8E 54657374696E67+  WBIO_M DC.B ’Testing word/byte ’,nul
000FA1 6F757470757473+  OUTP_M DC.B ’outputs ........ ’,nul
000FB2 696E7075747320+  INPT_M dc.b ’inputs ........ ’,nul
000FC3 73776170206F75+  SWPO_M DC.B ’swap out ...... ’,nul
000FD4 7377617020696E+  SWPI_M DC.B ’swap in ....... ’,nul
000FE5 7377617020492F+  SWIO_M DC.B ’swap I/O ...... ’,nul

000FF6 5061737365640D+  PASS_M DC.B ’Passed’,CR,LF,nul
000FFF 4661696C65640D+  FAIL_M DC.B ’Failed’,CR,LF,NUL

001008 0D0A0A50617373+  PCNT_M DC.B CR,LF,LF,’Passed ’,NUL
001013 2C204661696C65+  FCNT_M DC.B ’, Failed ’,NUL

00101D 0D0A5265636569+  NULL_M DC.B CR,LF,’Received NULL command.’,NUL
001036 0D0A5265636569+  ACKN_M DC.B CR,LF,’Received ACKN command.’,NUL

00104F 4661696C656420+  BADB_M DC.B ’Failed ==> Bad DATA from "slave"!’,CR,LF,NUL
001073 4661696C656420+  BADA_M DC.B ’Failed ==> Bad ACKN from "slave"!’,CR,LF,NUL
001097 4661696C656420+  BADD_M DC.B ’Failed ==> Bad DMAx from "slave"!’,CR,LF,NUL

0010BB 204164643A00     ADDR_M DC.B ’ Add:’,NUL
0010C1 2057723A00       WROT_M DC.B ’ Wr:’,NUL
0010C6 2052643A00       READ_M DC.B ’ Rd:’,NUL
0010CB 204353523A00     CHSR_M DC.B ’ CSR:’,NUL
0010D1 204253523A00     BSR_M  DC.B ’ BSR:’,NUL
0010D7 205354533A00     STAT_M DC.B ’ STS:’,NUL
0010DD 20464C473A00     FLAG_M DC.B ’ FLG:’,NUL

                        * dump tables to console messages

0010E3 0D0A0A72656769+  DT0     DC.B CR,LF,LF,’registers:  csr|dcr|ocr|amr|ema|ccr| mtc.w | mac.h | mac.l |dicr’
001126 0D0A2020202020+          DC.B CR,LF,’                0 | 1 | 2 | 3 | 4 | 5 | 6   7 | 8   9 | A   B | C  ’,NUL
001169 0D0A545820204C+  DT1     DC.B CR,LF,’TX  Long     $’,nul
001179 0D0A5458202057+  DT2     DC.B CR,LF,’TX  Word     $’,nul
001189 0D0A5458202042+  DT3     DC.B CR,LF,’TX  Byte     $’,nul
001199 0D0A525820204C+  DT4     DC.B CR,LF,’RX  Long     $’,nul
0011A9 0D0A5258202057+  DT5     DC.B CR,LF,’RX  Word     $’,nul
0011B9 0D0A5258202042+  DT6     DC.B CR,LF,’RX  Byte     $’,nul
0011C9 2C2400           DTL     DC.B ’,$’,NUL

                        **********************************************************************
                        ***                                                                ***
                        **          DR11WA Test Constants, Variables, & Storage             **
                        ***                                                                ***
                        **********************************************************************
0011CC 00               STATS    DC.B      0                board status register stg
0011CE 0000             COMMAND  DC.W      0                attn command word storage
                        *-------------------------------------------------------------------*
                        * Test Enables for individual tests     (Enabled = 1, Disabled = 0)
                        *
0011D0 FFFF             TEST     DC.W      $FFFF         Enable Flags          (all default)
                        *        bit       15                Master mode
                        *        bit       14                Slave mode
                        *        bit       13                Burst mode
                        *        bit       12                Cycle mode
                        *       bit       11              Long data
                        *       bit       10              Word data
                        *       bit        9              Byte data
                        *       bit        8              Swap word out
                        *       bit        7              Swap word in
                        *       bit        6              Swap byte out
                        *       bit        5              Swap byte in
                        *       bit        4              Swap word/byte out
                        *       bit        3              Swap word/byte in
                        *       bit        2              Swap word/byte out/in
                        *       bit        1              enable test data tables
                        *       bit        0              undf
                        *
                        *--------------------------------------------------------------*
                        * Slave mode command table
                        *
0011D2 000005B8         SCMDS   DC.L      NULLCMD         0    slave
0011D6 000005D4                 DC.L      ACKNCMD         1    master/slave
0011DA 000005C2                 DC.L      LINKCMD         2    slave
0011DE 000008A0                 DC.L      SETBYTE         3    slave
0011E2 00000876                 DC.L      SETWORD         4    both
0011E6 0000084C                 DC.L      SETLONG         5    both
0011EA 00000628                 DC.L      SLAVE_TX        6    slave
0011EE 00000682                 DC.L      SLAVE_RX        7    slave
0011F2 0000070C                 DC.L      SETBRST         8    both
0011F6 0000073E                 DC.L      SETCYCL         9    both
0011FA 000005E8                 DC.L      EXITCMD         A    slave
0011FE 00000602                 DC.L      ABRTCMD         B    slave
001202 0000061E                 DC.L      SWAPCMD         C    slave
001206 000005DE                 DC.L      BADCMD          D    master/slave
00120A 0000000D         VALID   DC.W      $0000,$000D
                        *--------------------------------------------------------------*
                        * registers:  csr|dcr|ocr|amr|ema|ccr| mtc.w | mac.h | mac.l |dicr
                        *              0 | 1 | 2 | 3 | 4 | 5 | 6   7 | 8   9 | A   B | C
00120E FF383239FB88FF+  TX.L  DC.B    $FF,$38,$32,$39,$FB,$88,$FF,$FF,$FB,$00,$00,$00,$3E
00121B FF381279FB88FF+  TX.W  DC.B    $FF,$38,$12,$79,$FB,$88,$FF,$FF,$FB,$00,$00,$00,$3E
001228 FF300279FB88FF+  TX.B  DC.B    $FF,$30,$02,$79,$FB,$88,$FF,$FF,$FB,$00,$00,$00,$3E
                        *                |   |   |   |   |   |       |       |       |
001235 FF38B239FB88FF+  RX.L  DC.B    $FF,$38,$B2,$39,$FB,$88,$FF,$FF,$FB,$00,$00,$00,$3E
001242 FF389279FB88FF+  RX.W  DC.B    $FF,$38,$92,$79,$FB,$88,$FF,$FF,$FB,$00,$00,$00,$3E
00124F FF308279FB88FF+  RX.B  DC.B    $FF,$30,$82,$79,$FB,$88,$FF,$FF,$FB,$00,$00,$00,$3E
                        *--------------------------------------------------------------*
                        * Transfer mode pointers
                        *
00125C 0000120E         TXS       DC.L      TX.L            default is long tx table
001260 00001235         RXS       DC.L      RX.L            default is long rx table
001264 000009D8         FBS       DC.L      FILL_L          def is long fill buff subr
001268 00000A8C         TBS       DC.L      TEST_L          def is long test buff subr
                        *--------------------------------------------------------------*
                        * DR11WA Test Date Table (TDATA)
                        *
00126C AAAAAAAA+        DTBL      DC.L      $AAAAAAAA,$55555555,$CCCCCCCC,$33333333
00127C 0F0F0F0F+                  DC.L      $0F0F0F0F,$F0F0F0F0,$00FF00FF,$FF00FF00
00128C 0000FFFF+                  DC.L      $0000FFFF,$FFFF0000,$FFFFFFFF,$00000000
00129C 0000126C         DPTR      DC.L      DTBL            test data table pointer
```

```
0012A0 00000000          TDATA    DC.L      0                     current test data variable
0012A4 00                DSIZE    DC.B      0                     current test data size
                         **********************************************************************
                                  END       DR11A4
 0x12A8 bytes code/data,  0x0 uninitialized
```